R E V I S T A   P A R A   U S U A R I O S   D E

Drean Commodore

Nº 3 ₳ 2.00   REP. ARGENTINA



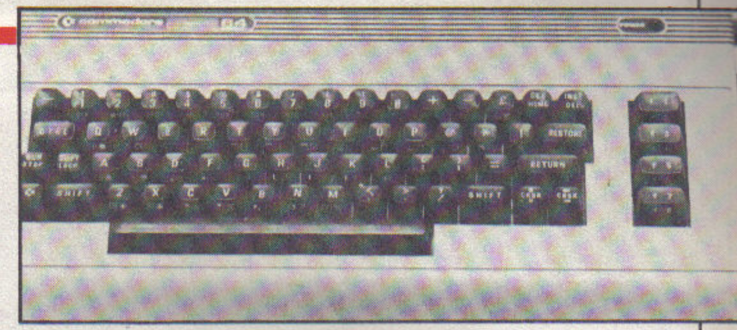
SUMARIO

NOTAS TECNICAS

PROGRAMAS

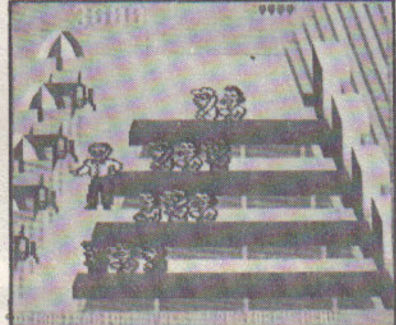
REVISION DE SOFTWARE

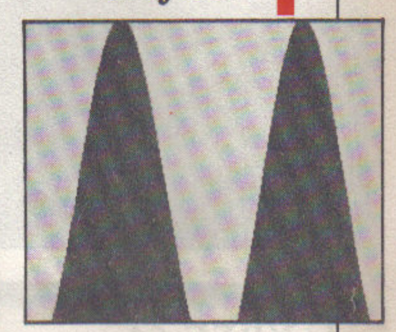
SECCIONES FIJAS



Funciones para que el intérprete Basic pueda ejecutar un programa determinado.

Continuamos explicando los comandos de la Drean Commodore 16 empezando esta vez con el Modo Gráfico.





Nuestras calificaciones para algunos de los mejores juegos disponibles en el mercado argentino.

Drean Commodore AÑO 1 - Nº 3

Director General
Ernesto del Castillo
Director Editorial
Cristian Pusso
Director Periodístico
Fernando Flores

Director Financiero
Javier Campos Malbran
Arte y Diagramación
Fernando Amengual
Coordinador
Ariel Testori

Redacción
Cristian Parodi
Departamento de Avisos
Oscar Devoto
Departamento de Publicidad
Guillermo González Aldanur

Drean Commodore es una Revista mensual editada por Editorial PROEDI S.A. (e./T.), Cerrito 1320, 1º Piso, Buenos Aires, Te.: 42-9681/9. Registro Nacional de la Propiedad Intelectual: E.T. M. registrada.
Queda hecho el depósito que indica la Ley 11.723 de Propiedad Intelectual.

Precio de este ejemplar: ₳ 2.
Impresión: Calcotam, Fotocromo tapa: Columbia, Fotocomposición: Van Waveren.
Los ejemplares atrasados se venderán al precio del último número en circulación.

Distribuidor en Capital: Martino, Juan de Garay 358, P.B. Capital. Distribuidor interior: DGP: Hipólito Yrigoyen 1450, Capital Federal. T.E. 38-9266/9800

Quick Data Drive



QUICK DATA DRIVE

La función de este dispositivo es permitirnos grabar y cargar programas rápidamente entre éste y la C-64 o VIC-20. Como ustedes saben cargar programas con el DATASSETTE o con un grabador común, demora bastante tiempo si no se dispone de un Turbo-Tape o de algún otro programa utilitario de este tipo. No sólo se demora con éstos; también se demora con el DISK DRIVE 1541. Por ejemplo para cargar un programa cuya extención es de 24 kb el DATASSETTE demora 8 minutos; a la 1541 le lleva 1 minuto efectuar esta tarea y al QUICK DATA DRIVE tarda nada menos que 20 segundos. El soporte magnético que utiliza éste es un cassette especial (denominado wafer) cuya cinta está unida en los extremos constituyendo una cinta sin fin y girando a gran velocidad. Antes de utilizar este soporte debemos cargar en la computadora el sistema operativo de el llamado QOS (Quick Operating Sistem. Luego de cargarlo el drive está listo para operar. Este dispositivo no está aún disponible en nuestro mercado.

XREF-64

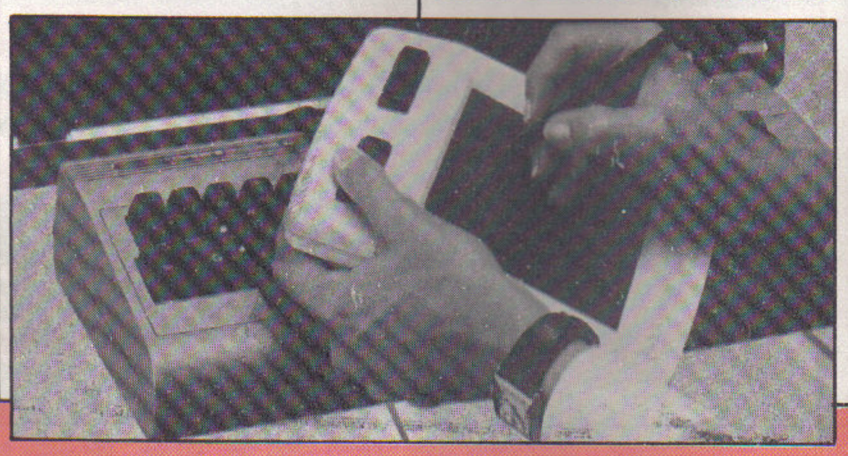
Es una herramienta indispensable para autores de programas Basic. Este utilitario puede proveer información al usuario como ser uso de todas las variables, números de líneas, constantes numéricas y de los comandos Basic que estemos utilizando. Nosotros podemos inmediatamente saber en qué número de línea se encuentra una variable o un comando Basic determinado. Este utilitario es, además, óptimo para tareas de debugging sobre nuestros programas. El autor de éste es Abacus Software; viene en diskette con su correspondiente manual explicativo.

ASSEMBLER

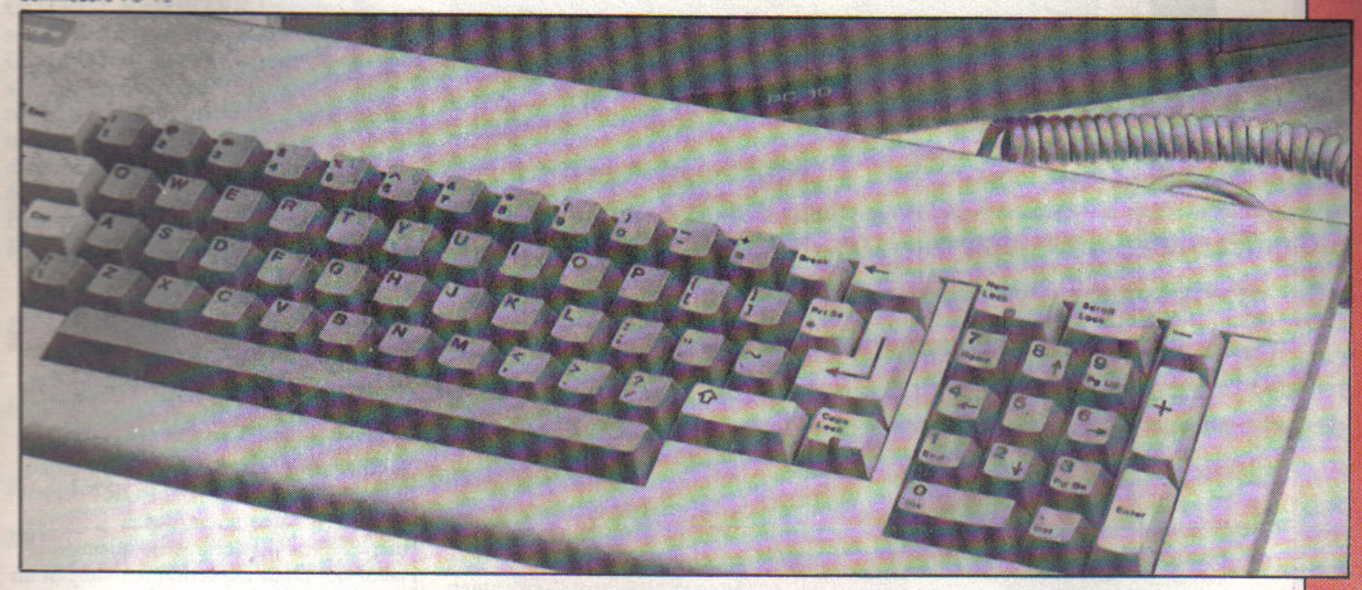
Este utilitario nos permite desarrollar programas en lenguaje máquina y/o assembler del C-64. Alguna de las características del assembler son:
Rápido macro assembler
Capacidad de ensamblaje condicional
Completo editor de pantalla para el programa fuente
Posibilidad de encadenar archivos fuentes
El monitor dispone de 15 funciones incluyendo:
Localización de caracteres dentro del programa
Transfiere blockes de datos
Compara blockes de datos
Accede a bancos de memoria determinados
Rápido trace con puntos de parada
Puede operar con el assembler.

CADPAK-64:

Este utilitario nos permite, entre otras cosas, poder dibujar sobre la pantalla con el lápiz óptico (este último es posible encontrarlo en nuestro mercado). Los gráficos se realizan en alta resolución, por lo que podemos hacer desde simples dibujos hasta cuadros artísticos. Luego, a través de una rutina hardcopy, podemos imprimir todo lo que editamos a través de impresoras series Commodore, Okidata, Epson series RX, MX o FX. El manejo de CADPAK-64 es muy sencillo. El menú principal nos permite



Commodore PC-10



COMMODORE PC-10

Este es otro de los desarrollos de Commodore: su Personal Computer PC-10. Debido a su bajo costo y sus grandes características se ha convertido en una estupenda PC pudiendo competir con sus similares de IBM, APPLE. Básicamente se compone de:
MICROPROCESADOR: INTEL 8086 (Procesador de 16 bits)
INTEL (8087) (Procesador coma flotante)
CLOCK: 4.77 Mhz
MEMORIA: 256 kb ampliable a 640 kb
INTERFACE: Port paralelo (Centronics) Port serie (RS 232) 5 slots de expansión
ALMACENAMIENTO: Unidad doble de discos flexibles de doble cara 360 kb Floppi de 360 kb.
Un disco duro de 10 Mb
SIST. OPERATIVO: DOS 2.11
BIOS: Rom de 8kb para manejo de interrupciones, entrada/salida y del bootstrap
DOS: Carga automática desde el disco. Manejo de archivos e interpretador de comandos.

seleccionar los diferentes items con sólo el lápiz óptico, el item buscaddo sobre la pantalla. Es decir que el uso del teclado es mínimo.
Otra de las características principales es su editor de "objetos". A través de él podemos definir la forma de objetos tales como circuitos electrónicos, maquinarias, etc. Estas definiciones pueden ser tan complejas como la resolución de la pantalla lo permita. Luego es posible guardarlas en diskette para formar una especie de "biblioteca".
De esta forma es fácil volver a utilizarlas pudiendo, si es necesario, variar su escala o su rotación.

PILOT PARA LA C-64

Pilot es un lenguaje intérprete de alto nivel. Dispone de graficos en alta resolución. La forma de crearlos es a través de una "tortuga" que acepta nuestras órdenes tales como desplazarse a la izquierda, rotar, adelantar un paso, etc. (muy similar a la del lenguaje LOGO). El lenguaje en si es, en algunos aspectos, similar al último citado. El manejo de SPRITES se realiza en forma muy sencilla debido a un completo editor de sprite.

CP/M PARA LA C-64

Para aquellos que no lo conocen, CP/M es el sistema operativo mas difundido para computadoras que tienen como unidad central de proceso el microprocesador Z80, 8080 u 8085 (todos ellos de 8 bits). Hablar de CP/M implica hablar de más de 100000 usuarios en todo el mundo y, también, de más de 1000 paquetes de software. La C-64 permite el ingreso de este sistema operativo. Para ello se necesita insertar una tarjeta (plaqueta) la cual contiene el micro Z80 junto con la memoria adicional ya que este sistema operativo requiere de, por lo menos, 48 kb de memoria libre.
También se deben efectuar una serie de cambios dentro del mapa de memoria del C-64. Esto se debe a que CP/M utiliza otra configuración de memoria. Lamentablemente tanto la plaqueta como el correspondiente software no se encuentran disponibles en nuestro mercado.
Esperemos que en un futuro muy cercano la empresa DREAN desarrolle este sistema operativo juntamente con algún compilador como ser algo parecido al CBASIC de MICROSOFT, FORTRAN o COBOL.

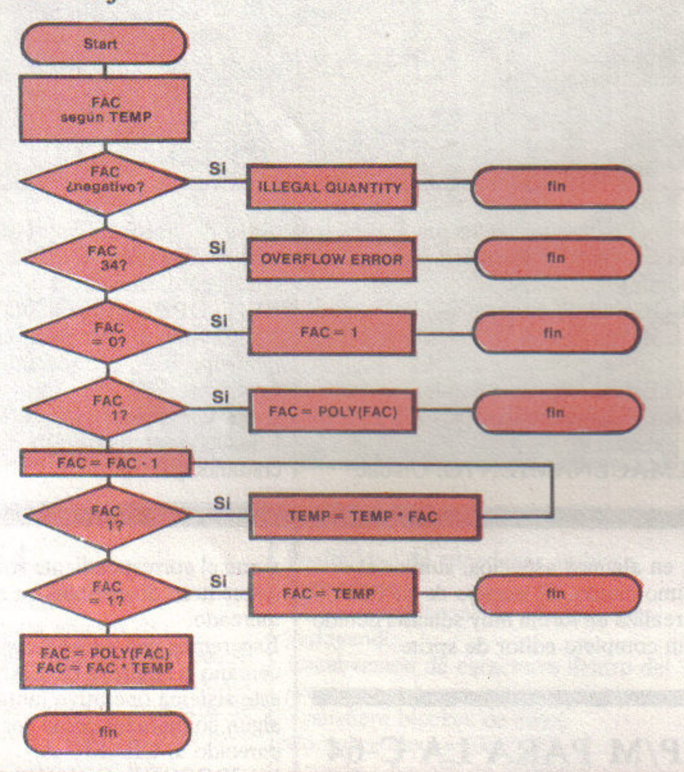
SEGURIDAD

A través de una C-64 juntamente con sensores y diodos emisores de luz infrarrojo, se ha desarrollado en Estados Unidos un sistema de alarma antirrobo y anti-incendio. De esta manera cualquier intruso que ingrese a nuestra casa será detectado por la C-64. De igual forma cualquier aumento de temperatura que sobrepase el valor normal será rapidamente detectado. El sistema está formado por la interface correspondiente que se inserta al Port del usuario y del correspondiente programa que administra y controla a esta.

MAPA DE MEMORIA DEL C-64

Funciones para que el intérprete Basic pueda ejecutar un programa determinado.

Diagrama de flujo.



A partir de este número explicaremos las direcciones de memoria del C-64. Cada una de éstas excepto las que constituyen el área de memoria libre, posee una determinada función para que el intérprete Basic pueda ejecutar un programa determinado. Es decir que, conociéndolas, podremos desarrollar programas que utilicen sus funciones logrando así mayor velocidad (por supuesto escribiendo el programa en código de máquina). Comenzaremos describiendo las direcciones pertenecientes a la:

PAGINA CERO

Cada página de memoria en la C-64 está formada por 256 Bytes. Esta, por ser la primera, está formada por las direcciones que van desde la $0000 hasta la $00FF. Esta página puede considerarse la más importante dentro del computador. Hemos visto que la principal ventaja de trabajar con ésta (almacenar-recuperar información) es la rapidez con que se ejecutan las instrucciones respectivas.

La dirección $0000 y la $0001 (las dos primeras) tienen una especial función en I/0 (Input/Output-Entrada/Salida). En el caso de la Drean Commodore 64 éstas determinan la disposición de toda la memoria.

DIRECCION $0000 (0 DECIMAL)

Esta dirección representa a un registro denominado DATA DIRECTION REGISTER (registro de dirección de datos). Cada Bit de este registro determina si el contenido del correspondiente Bit en la I/0 PORT interna (PERIPHERAL INTERFACE BUFFER-interface de buffer periférico) será utilizado como entrada o salida. El I/0 PORT transfiere datos hacia/desde dispositivos periféricos de acuerdo al valor del Data Direction register. Si un determinado bit de este registro es "activado" con el valor "0", indicará que la dirección del dato es de entrada, es decir que el Bit del I/0 PORT podrá ser afectado por el dispositivo periférico. Si el Bit es puesto en "1", la dirección de los datos será de salida (ese determinado Bit será utilizado como salida).

Los únicos Bits significativos de este registro son el 0,1,2,3,4,5.

El Bit 6 y 7 no tienen uso. Cuando prendemos la computadora, el contenido de este registro es de 239 ($EF), lo que indica que todos los Bits, excepto el número 4, están activados

Constitución del DATA DIRECTION REGISTER

Bit 0: Dirección del Bit 0 en el I/O PORT (por Default se asume "1", es decir salida)

Bit 1: Dirección del Bit 1 en el I/O PORT (por Default se asume "1", es decir salida)

Bit 2: Dirección del Bit 2 en el I/O PORT (Por Default se asume "1", es decir salida)

Bit 3: Dirección del Bit 3 en el I/O PORT (Por Default se asume "1", es decir salida)

Bit 4: Dirección del Bit 4 en el I/O PORT (Por Default se asume "0", es decir entrada)

Bit 5: Dirección del Bit 5 en el I/O PORT (Por Default se asume "1", es decir salida)

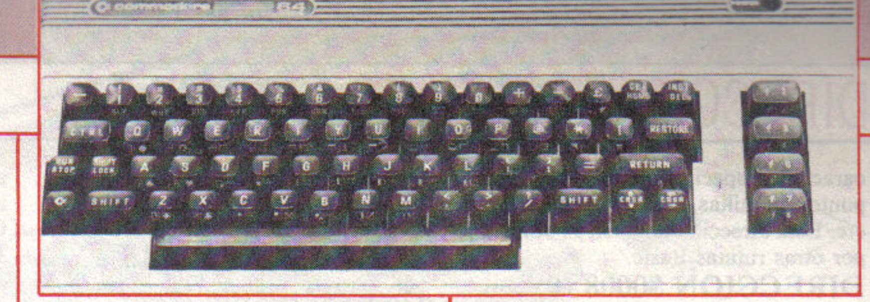
Bit 6: Dirección del Bit 6 en el I/O PORT. No se usa.

Bit 7: Dirección del Bit 7 en el I/O PORT. No se usa.

DIRECCION $0001 (1)

Esta dirección representa a un registro denominado PERIPHERAL

DIRECCIONES



OUTPUT REGISTER (registro de salida periférico, más conocido como OUTPUT REGISTER-registro de salida). La Principal función de este registro es determinar cuáles bloques de memoria RAM o ROM serán direccionados por el microprocesador 6510.

Este registro está formado por:

Bit 0: LORAM. Selecciona ROM o RAM en 40960 ($A000). 1=BASIC, O=RAM.

Bit 1: HIRAM. Selecciona ROM o RAM en 57334 ($E000). 1=Kernal, O=RAM.

Bit 2: CHAREN. Selecciona caracteres en ROM o dispositivos de I/O. 1=I/O, O=ROM.

Bit 3: Línea de salida de datos del cassette.

Bit 4: Sentido del switch del cassette. Lee O si el botón fue presionado, 1 si no fue.

Bit 5: Control del motor del cassette.

Bit 6-7: No se utilizan.

El significado de estos Bits es el siguiente:

Bit 0: Este Bit controla la LORAM (LOWRAM - RAM baja). Con un "0" en este Bit se conmuta la BASIC ROM. Esta es reemplazada por RAM a partir de las direcciones 40960-49151 ($A000-$BFFF). Si se omite, el sistema operativo asume "1".

Bit 1: Este Bit controla el HIRAM (HIGH RAM). Con un "0" se conmuta la ROM del Kernal (sistema operativo de C-64). Esta se reemplaza por memoria RAM a partir de la dirección 57344 hasta la 65535 ($E000-$FFF). Como el intérprete Basic utiliza el Kernal, éste también es conmutado y sustituido por memoria RAM. Desde ya, todo intento por querer desarrollar algún programa en Basic, con el Kernal desactivado, será en vano. Si no se especifica el valor de este Bit, por default se asume "1".

Bit 2: Este Bit controla el CHAREN. Con un "0" en esta posición se activa el generador de caracteres en ROM. De esta forma pueden ser leídos por el 6510 en las direcciones 53248-57343 ($D000-$DFFF). Normalmente este Bit es puesto en "1" para que mientras el VIC accede al generador con propósitos de crear una pantalla determinada, el usuario no pueda efectuar el comando PEEK dentro de ésta. Como esta ROM es conmutada dentro de las mismas posiciones que ocupa el I/O, no puede ocurrir ningún proceso de entrada/salida.

Bit 3: Este indica la línea de datos de salida hacia el cassette. Esta es conectada a la línea de escritura del cassette en la Port correspondiente. Se utiliza para enviar los datos que se almacenaran en cinta.

Bit 4: Indica la función que se está realizando en el DATASSETTE. Un "1" indica que se oprimió la tecla de grabación.

Bit 5: Este controla el motor del DATASSETTE. Si ponemos un "0" el motor seguirá dando vueltas cuando presionemos uno de los botones del grabador.

DIRECCION $0002 (2)

Esta dirección no tiene un uso específico. Podemos utilizarla cuando debamos escribir algún dato o el estado de alguna variable. En más de una ocasión debemos usar, en nuestro programa, variables que trabajen como Flag, indicadores; sólo toman el valor de 1 ó 0 y con ello nosotros determinamos la acción a seguir. Si, por algún motivo, debemos disponer de 8 Flags distintos, necesitaríamos 8 variables, lo que implicaría un consumo de memoria. Si utilizamos una determinada dirección de memoria la cual sabemos que no puede ser modificada por el intérprete Basic o por el Kernal, como la $0002, nos ahorraríamos todas esas variables ya que con cada Bit de esta dirección podemos representar los diferentes indicadores. De ante mano determinamos que, por ejemplo, el Bit 0 representa un ingreso erroneo (si es "1" el ingreso fue erróneo si es "0" fue correcto); el Bit 1 indica si hubo (con "1") exceso en la longitud de los registros o (con "0") si no lo hubo; etc. Además recuerden que por pertenecer esta dirección a página cero, todas las instrucciones de carga y almacenaje en ella se ejecutan mucho más rápido que en otras.

DIRECCIONES $0003-$0004 (3-4)

Antes de describir estas direcciones recordemos dos cosas:

1) Se dice que una dirección de memoria es un "vector" cuando los contenidos de éstas apuntan a la dirección verdadera de la rutina determinada.

2) El formato que utiliza el 6510 para almacenar una dirección es Byte bajo y luego el Byte alto

Esta dirección es un vector. Los contenidos apuntan a la dirección de la rutina que convierte un número de coma flotante a entero. La dirección de ésta es $B1AA (si listamos los contenidos de las direcciones $0003 y $0004 comprobaremos que éstas son $AA y $B1 respectivamente).

DIRECCIONES $0005-$0006 (5-6)

Estas direcciones trabajan como un vector. Sus contenidos apuntan a la dirección que posee la rutina que convierte números enteros a números en coma flotante. Esta se encuentra en la dirección $B391 (45962).

DIRECCION $0007 (7)

En esta dirección el intérprete Basic almacena los códigos ASCII de

caracteres especiales tales como los dos puntos, comillas, terminador de línea, etc. Esta dirección también es utilizada por otras rutinas Basic.

DIRECCION $0008 (8)

En esta dirección se utiliza para poner los códigos de fin de línea (0) o para poner el código de comillas (34). El intérprete Basic usa esta dirección durante la "tokenización" del texto Basic.

DIRECCION $0009 (9)

El contenido de esta dirección determina la columna donde se encuentra el cursor antes del ultimo TAB o SPC. Este valor es seteado desde la dirección $D3 (211). Se utiliza para calcular en dónde finaliza el cursor cuando alguna de estas funciones son invocados. Comprobrarán que el valor que tendrá esta dirección estará comprendida entre 0-79. Estos representan la posición del cursor sobre una línea lógica. Cada línea lógica está formada por dos líneas físicas (0-39).

DIRECCION $000A (10)

Esta dirección se utiliza como FLAG. El Basic utiliza una rutina del Kernal para realizar el LOAD o el VERIFY. La realización de alguna de ellas dependerá del valor que tendrá el acumulador antes de ingresar a esta rutina. Si es "0" se efectuará la primera, con "1" la segunda. El intérprete Basic pone el correspondiente valor en esta dirección de acuerdo al comando que se esté ejecutando.

DIRECCION $000B (11)

La rutina que convierte el texto Basic en el buffer de entrada en líneas de programa ejecutables (conocidas como "tokens") y la que encadena esas líneas utilizan esta dirección como un indice dentro del buffer de entrada. Cuando culmina el trabajo de convertir el texto en "tokens", esta dirección contiene la longitud de la línea "tokenizada".
La rutina que crea los vectores (array) o localiza un elemento en un conjunto utiliza esta dirección para calcular el número de DIMensions invocados para calcular la cantidad de almacenamiento requerido cuando se encuentra una nueva sentencia DIM. También se utiliza para calcular el número de elementos de un conjunto cuando se requiere de alguno de ellos.

DIRECCION $000C (12)

Esta dirección es utilizada como Flag por la rutina que crea o localiza un array. Se utiliza para saber si una variable es una array; si éste fue dimensionado o si un nuevo array se

Conociendo las direcciones de memoria podremos ampliar el Basic residente en la C-64

dimensiona por default (sería hacer DIM (10)

DIRECCION $000D (13)

Esta dirección se utiliza para saber cuándo un determinado dato es numérico o string. Si el contenido de esta dirección es $FF (255) indicará que se trata de una variable string mientras que un 0 indicará que es una variable numérica. Esto se realiza cada vez que se crea una nueva variable.

DIRECCION $000E (14)

Esta dirección se utiliza como flag para determinar qué tipo de dato numérico está siendo utilizado por el Basic (Punto flotante o entero). Si el contenido de esta dirección es $80 (128) indicará que se trata de un número entero mientras que con un 0 será punto flotante (variable real)

DIRECCION $000F (15)

El contenido de esta dirección es utilizadda por el proceso de conversión de una línea de texto Basic en un determinado "token".
La rutina de la sentencia LIST utiliza el contenido de esta dirección como un flag para saber cuándo un caracter contiene comillas.

DIRECCION $0010 (16)

Esta dirección es utilizada por la rutina que busca o crea una variable y cheque si el nombre de una variable es valida. Si se encuentra un paréntesis abierto, este flag se setea para indicar que la variable en cuestión es un array o una función definida por el usuario.

DIRECCION $0011 (17)

El contenido de esta dirección es utilizado para indicarle al Basic si se está ejecutando una sentencia GET, READ o INPUT. Si el valor es $98 implicará que se trata de una READ; $40 GET; 0 INPUT. De esta manera el intérprete ejecutará las distintas rutinas acorde a la sentencia leída. Es decir que imprimirá el signo de interrogación cuando se ejecute un INPUT; no imprimirá nada cuando se ejecute un GET.
También se utiliza para indicar el respectivo mensaje de error.

DIRECCIÓN $0012 (18)

Esta dirección se utiliza para saber si el signo de la función SIN o TAN fue positivo o negativo.
La rutina de comparación de string y números utiliza esta dirección para indicar el resultado de la misma. Por ejemplo si se compara la variable A con la variable B habrá un "1" si A > B, 2 si A=B, 4 si A < B. Si el operador es una combinación de los anteriores, entonces el resultado también sera una combinación de los respectivos valores de comparación.

DIRECCION $0013 (19)

Cuando el Basic necesita una entrada o salida, él mira el contenido de esta dirección para determinar qué dispositivo se utilizará para tal propósito. También utiliza la dirección $B (184) para determinar cuál dispositivo está habilitado como entrada o como salida. Un 0 en ésta de dirección indicara que la entrada y salida son las normales (dispositivo 0 teclado, dispositivo 3 pantalla).
Cuando se utiliza otro dispositivo, el número lógico de canal es puesto en esta dirección.

DIRECCION $0014 - $0015 (20-21)

El número de "referencia" de las sentencias LIST, GOTO, GOSUB es almacenado aqui en el formato de Byte bajo-Byte alto.
El comando LIST almacena el número de línea más alto a listar (65535, $FFFF, si se lista hasta el final del programa)
El comando GOTO chequea si la línea la cual hay que saltar es mayor que el número de línea que actualmente se está ejecutando. Si es mayor el GOTO comienza la búsqueda del número de línea de salto a partir de la actual. Si no es mayor, el GOTO buscar la línea de salto a partir de la primer línea del programa.
Los comandos POKE, PEEK, WAIT y SYS utilizan esta dirección como puntero a la dirección en donde se pide tal comando.

DIRECCION $0016 (22)

Esta dirección apunta al próximo espacio disponible dentro del área del stack temporario para string ($19-$21). Si el contenido de esta dirección es $19 indicará que este stack está vacío, si es $1C no hubo ninguna entrada, si es $1F hubo dos entradas y si es $22, el stack está lleno.

INGRESO IRRESTRICTO A LA COMPUTACIÓN.

A partir de hoy, Commodore entra a todas las aulas.
Con Proceda.

Ahora, la computación es la mejor escuela.
Con Proceda. Y su exclusiva distribución del
computador Commodore en el área educativa.
Sumando a estos avanzados equipos, el más desarrollado software
y un constante servicio de mantenimiento y actualización.
Con esta designación, que nos enorgullece por su gran trascendencia,
todos los estudiantes podrán ingresar al futuro.
Haciéndolo realidad hoy. Con Drean Commodore y Proceda.



Informática Integral

Drean
commodore

Centro Especializado en Computación Personal:
Av. Córdoba 650 (Casi Florida) - Tel. 392-7611/8478
Casa Central: Av. Pueyrredón 1770. Tel. 821-2051

Sucursal Córdoba: Boulevard Reconquista 178. Tel. 36-207 y 39-520.
Centro: Peatonal San Martín 149 (Córdoba). Tel. 24-447.

INSTRUCCIONES PROPIAS

Continuamos explicando los comandos de esta máquina empezando esta vez con el Modo Gráfico.

Modo gráfico

Para poder utilizar los comandos "gráficos" de la C-16 se debe, antes que nada, ingresar al modo gráfico. Esto se realiza a través de la sentencia:

GRAPHIC modo,clear

donde: modo=0,1,2,3,4 siendo

0 Texto
1 Gráficos en alta resolución
2 Texto y gráficos en alta resolución
3 Gráficos multicolores
4 Texto y gráficos multicolores

clear=0,1 siendo

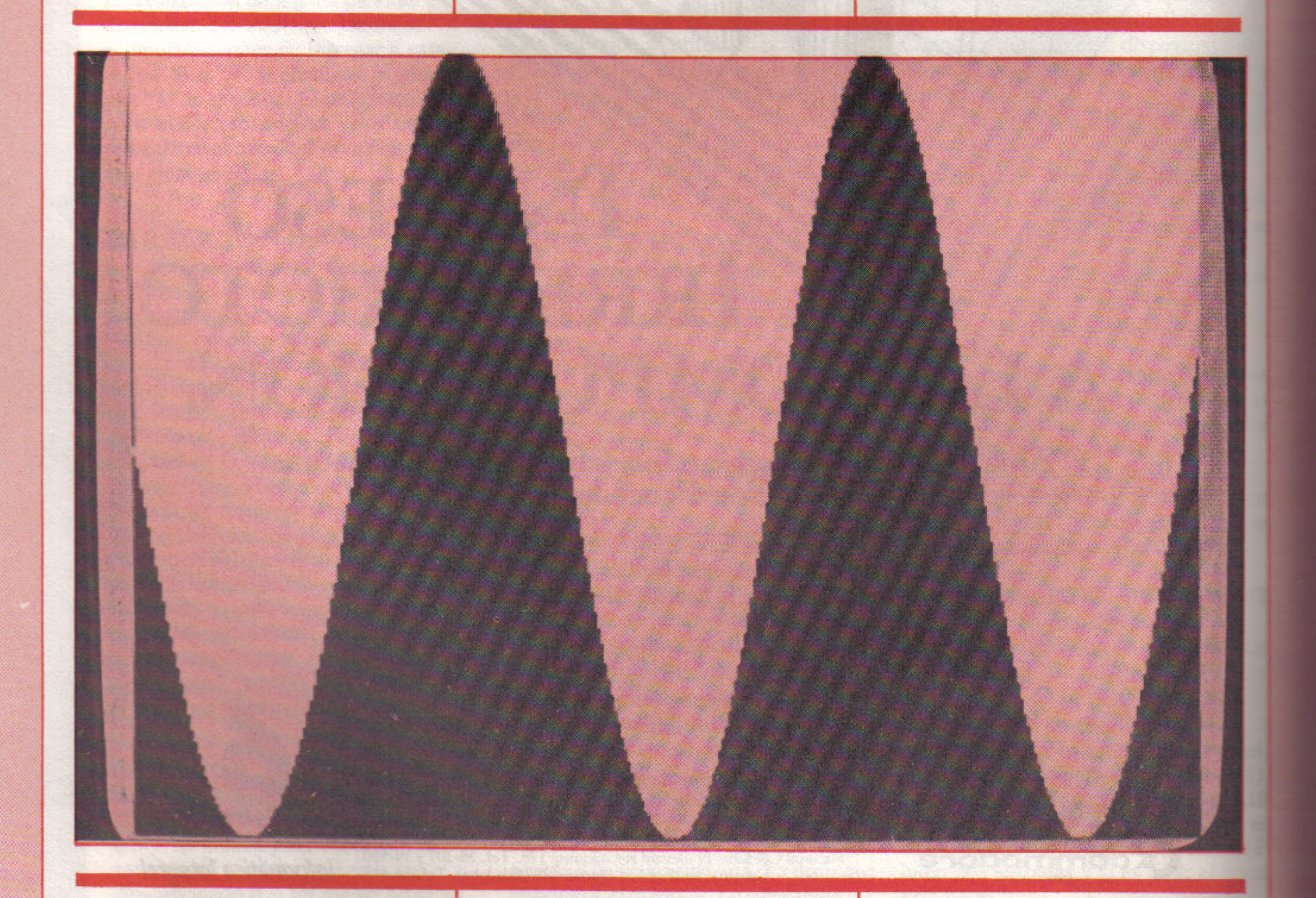
0 No se limpia la pantalla
1 Se limpia la pantalla (este parámetro es opcional. Si se omite se asume 0)

Con este comando podremos realizar todo tipo de gráficos.
Posibilita, además, hacer en un sector de la pantalla un gráfico en alta resolución con texto en la otra parte de ella.

Gráficos en alta resolución

La pantalla de la C-16 contiene un total de 25 líneas con 40 caracteres por línea. De esta forma se dispone de 1000 caracteres sobre la pantalla. Además, cada caracter está formado por una grilla de 8x8 (también se los conoce con el nombre de DOT). Se dispone así de una resolución máxima de 320x200 puntos, es decir de 64000 Dots. La C-16 puede manejar en forma individual cada uno de esos puntos.
En uno de los modos gráficos (modo 1) existe una limitación respecto al color de los caracteres. Sólo es posible usar dos colores. Estos son color del caracter y color de fondo. En el modo 3 es posible definir caracteres que contengan hasta cuatro colores.
El comando DRAW nos permite definir en la pantalla un punto, una línea o una forma dada. El formato de éste puede ser:

COMANDO
DRAW cf,c,f
DRAW cf,c,f TO c,f



DRAW cf TO c,f

EFECTO
PUNTO
LINEA
REALIZA UNA LINEA DESDE EL ULTIMO PUNTO HASTA EL ACTUAL

donde cf: color fuente
c : columna
f : fila

El primer parámetro indica si se dibuja un punto (valor 1) o si se borra (valor 0).
Por ejemplo, si queremos dibujar un punto en la columna 15 fila 15 hacemos:
DRAW 1,15,15
Para borrarlo:
DRAW 0,15,15
Para dibujar una línea se utiliza el segundo formato del DRAW.
De esta forma si deseamos dibujar una línea desde el punto 10,3 hasta el punto 22,12 hacemos:
DRAW 1,10,3 TO 22,12
Para borrarlo cambiamos el 1 por el 0 y efectuamos el mismo comando DRAW.
Ahora si realizamos:
DRAW 1 TO 150,100
dibujará una línea desde el último punto realizado hasta el punto 150,100.
La C-16 dispone, también, del comando CHAR. Con él podemos imprimir un mensaje en cualquier parte de la pantalla. Su formato es:
CHAR cf,c,f,"mensaje"
Por ejemplo si deseamos imprimir COMMODORE en la columna 10 fila 14 hacemos:
CHAR 1,10,14,"COMMODORE"
Algo importante de resaltar es que las columnas y filas de este comando están comprendidas entre 0 y 40; 0 y 24.
Otro de los comandos es BOX. Con él podemos dibujar cuadrados con sólo especificar los ángulos inferiores y superiores de ella. Su formato es:
BOX cf,c1,f1,c2,f2
Por ejemplo el comando:
BOX 1,10,10,100,100
dibujará un cuadrado desde el punto 10,10 hasta el 100,100. Es posible especificar en este comando un sexto parámetro. Este indicará el ángulo a rotar. Los formatos del comando BOX son:

COMANDO	EFECTO
BOX 1,c1,f1,c2,f2	CUADRADO
BOX 1,c1,f1,c2,f2,an	ROTA
BOX 0,c1,f1,c2,f2	BORRA UN AREA DE PANTALLA
BOX 1,c1,f1,c2,f2,,fill	CUADRADO RELLENO

donde c1,c2,f1,f2,an representan columna 1, columna 2, fila 1, fila 2 y ángulo respectivamente.
Si, en cambio, deseamos realizar círculos, debemos utilizar el comando CIRCLE. El formato de éste es:

COMANDO	EFECTO
CIRCLE 1,cc,cl,ra	CIRCULO
CIRCLE 1,col,fil,anc,al	ELIPSE
CIRCLE 1,col,fil,anc,co,fi	ARCO
CIRCLE 1,col,fil,anc,al,,an	ELIPSE TRASLADADA
CIRCLE 1,col,fil,anc,,,pan	POLIGONO

donde cc,cl es la coordenada del centro del círculo
col,fil valores normales de columna y fila
anc,al ancho y alto
an,pan ángulo y punto de ángulo

DREAN COMMODORE 16

Utilizando este comando también podemos realizar triángulos, cuadrados polígonos, etc. por ejemplo efectuando: CIRCLE 1,160, 100, 50,42,,,, 120 realizaremos un triángulo cuyos lados tienen 120 grados (externos).
Otro comando muy potente de la Drean Commodore 16 es el ~~PRINT~~. PAINT
Este se realiza para llenar una determinada área encerrada de un gráfico con un determinado color. Su formato es:
PAINT ~~PRINT~~ cf, c, f
donde cf: color fuente
c : columna
f : fila
Los parámetros c y f determinan un punt interno dentro de la figura a rellenar.
La C-16, como ya lo dijimos, tiene la posibilidad de trabajar con gráficos multicolores. En alta resolución, la C 16 nos permite controlar cada uno de los "pixel" (el punto más pequeño de la pantalla). Lamentablemente la asignación de colores a cada uno de los DOTS (8 x 8 pixels) en este modo es limitada. Sólo podemos especificar el color del carácter y el color de fondo. Trabajando en el modo multicolor es posible definir hasta cuatro colores por caracter. Para ello debemos setear correctamente los parámetros del comando GRAPHIC.
Para trabajar con gráficos multicolores sin texto se debe ingresar GRAPHIC 3 y GRAPHIC 4 para gráficos multicolores con cinco líneas de texto.
Pasemos ahora a los comandos de depuración y ejecución que dispone el programador.
Se encuentra el comando AUTO el cual enumera las líneas de nuestro programa automáticamente con el incremento seleccionado. Su formato es:
AUTO Inc
donde inc: incremento seleccionado.
Cada vez que oprimimos RETURN al finalizar una línea de programa, el comando genera la siguiente posicionando el cursor a continuación de ésta. Para irnos de este modo simplemente basta con ingresar AUTO sin ningún parámetro. Esta es la única forma de cancelar este mando.
El comando BACKUP se utiliza para efectuar copias en unidades duales de Disk Drive. Este comando copia todos los archivos de una unidad hacia el otro.
Su formato es:
BACKUP D ni TO D n2 [, ON U n]
donde n1: Drive emisor
n2: Drive receptor
n: número de unidad (este parámetro es opcional)
Un ejemplo de este comando sería BACK UP DDO TO D1, ON U9. Esto significa que se copiarán los archivos

```
LIST

10 COLOR0,1
20 COLOR1,2
30 GRAPHIC 1,1
40 LOCATE0,100
50 FORX=1TO319
60 Y=INT(100+99*SIN(X/20))
65 Y1=-INT(100+99*SIN(X/20))
70 DRAW 1 TO X,Y
75 DRAW 1 TO X,Y1
80 NEXTX
90 GETA$:IFA$=""THEN90
100 GRAPHIC 0

READY.
```

desde el drive 0 hacia el drive 1 en el disk drive número 9.
Otro comando similar a éste es el COPY. Su formato es:
COPY [D n1,] "npf1" TO [D n2,] "npf2" [,ON U n]
donde npf1: nombre programa fuente 1
npf2: nombre programa fuente 2
n: número de unidad
Este comando copia en el drive n2 con el nombre seleccionado en npf2 el programa almacenado en el n1 con el nombre de npf1.
Puede suceder quee npf1 sea igual a npf2. Un ejemplo sería pues:
COPY Do, "TAREA" TO D1, "TAREA. 1".
De esta forma el programa denominado TAREA en el Drive 0 se copiará con el nombre de TAREA. 1 en el Drive 1.
El comando DELETE nos permite borrar una serie de líneas en forma rápida. Este nos evita la molesta tarea de borrar línea por línea. Su formato es:
DELETE li-lf
donde li: línea inicial
LF: línea final
Es decir que para borrar, por ejemplo, las líneas que van desde la 10 hasta la 100 simplemente hacemos:
DELETE 10-100
El comando DIRECTORY imprime el directorio del diskett actual sobre el drive y unidad seleccionado. Su formato ya lo hemos explicado en el número anterior.

Con el comando KEY podemos asignar a cada una de las ocho teclas de función los comandos que con más frecuencia utilizamos. Su formato es: KEY n, co.
donde n: número de la tecla de función
co: comando válido de la C-16
Por ejemplo si deseamos asignar el comando LIST a la tecla de función F7 se debe realizar:
KEY 7, "LIST" + CHR$ (13)
El CHR$ (13) es el código de RETURN. Es decir que cuando oprimamos la tecla F7 se ejecutará el mando LIST. Si no hubiésemos puesto ese código tendríamos que oprimir nosotros mismos la tecla de RETURN para que este se ejecutase.
El comando RENUMBER nos permite re-enumerar las líneas de nuestro programa. Su formato es:
RENUMBER ni, inc, lc
donde ni: número de línea inicial
inc: incremento
lc: línea de comienzo
Con el parámetro ni seleccionamos la nueva línea de inicio de nuestro programa. Con inc el incremento que habrá entre las líneas de éste.
Finalmente con lc determinamos a partir de qué línea deseamos la renumeración. Un ejemplo de ello sería:
RENUMBER 1000, 5, 100
Aquí le dedicamos al sistema operativo que enumeraremos las líneas de nuestro programa a partir de la línea 100, cambiando ésta por el número 1000 y realizando incrementos de a 5.

PROGRAMACION ESTRUCTURADA

Presentamos ahora algunos comandos orientados a la depuración de programas y manejo de las teclas.

En el número anterior comenzamos a describir las ventajas y desventajas del SIMONS BASIC. Presentamos ahora algunos comandos orientados a la depuración de programas y manejo de las teclas de función.

Describiremos las sentencias que este Basic Plus posee para la programación estructurada.

Sentencia: IF-THEN-ELSE
Formato: IF condición THEN sentencia A:ELSE: sentencia B
Función: Ejecuta la/s sentencia/s representada/s por A si la condición es verdadera. Caso contrario (condición falsa) ejecuta la/s sentencia/s B.

Noten los dos puntos (:) que separan el ELSE de la sentencia. Estos son obligatorios y en caso de no ponerse el intérprete entenderá que el comando ELSE debe ejecutarse si la condición es verdadera. Pero éste no existe como comando, por lo que se interrumpirá la ejecución del programa actual y se imprimirá el correspondiente error de sintaxis.

Sentencia: REPEAT-UNTIL
Formato: REPEAT sentencias del loop... UNTIL condición.
Función: Ejecuta una serie de sentencias hasta que la condición del UNTIL sea satisfecha.

Cuando, en el Basic común del C-64, deseamos realizar una serie de sentencias hasta que una condición sea satisfecha debemos utilizar el IF-THEN con los respectivos GOTO. Aquí, simplemente, indicamos de dónde a dónde abarca ese grupo de sentencias con REPEAT y UNTIL juntamente con la condición determinada. Como ejemplo analicen el siguiente:

```
10 REPEAT
20 AL=INT(RND(0)*100)+1
30 PRINT AL
40 UNTIL AL=54 OR AL=45
50 END
```

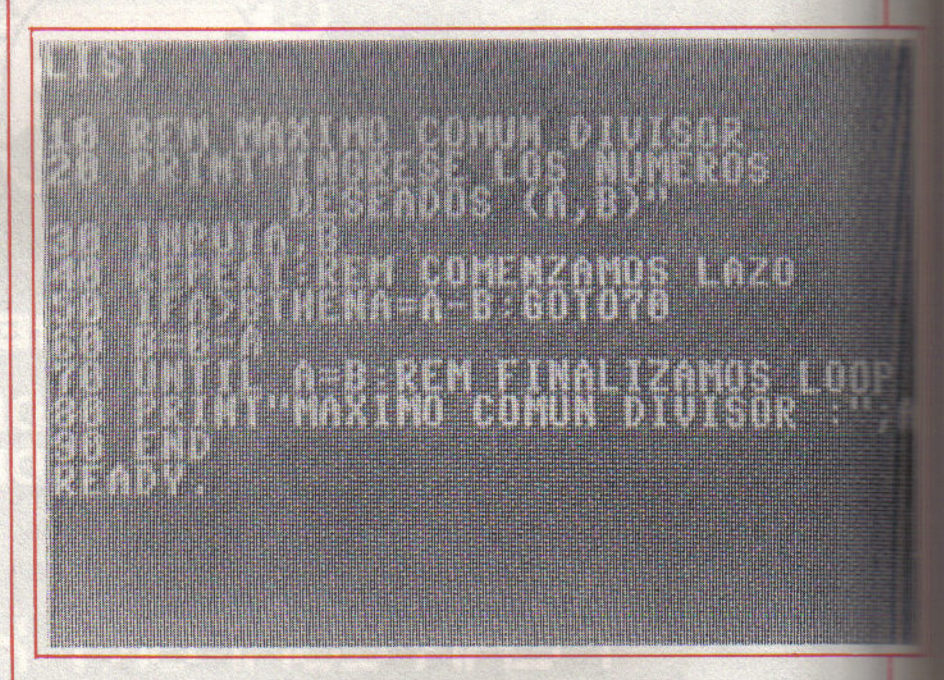
Primero se setea en la variable AL un valor aleatorio comprendido entre 1 y 100. Se imprime. Si ese valor es distinto de 54 o de 45, automáticamente se comienza el lazo nuevamente. Caso contrario se finaliza la ejecución. Habrán notado que, por lo menos, el lazo se ejecuta una vez ya que el predicado evaluativo se encuentra al final y para llegar a él se deben ejecutar las sentencias anteriores. Esto se debe tener presente cuando deseamos ejecutar un número de sentencias mientras la condición sea falsa.

Sentencia: LOOP-EXIT IF-END LOOP
Formato: LOOP sentencias ejecutables... EXIT IF condición... END LOOP
Función: Ejecuta un grupo de sentencias hasta que la condición sea verdadera.

La diferencia de ésta con la anterior es que el predicado evaluativo se encuentra en cualquier sector del lazo y no al final. Un ejemplo sería:

```
10 INPUT "INGRESE UN NUMERO DIGITO POR DIGITO"
20 LOOP
30 GET NUM$
40 A=ASC(NUM$)
50 EXIT IF A<48 OR A>57
60 PRINT NUM$;
70 END LOOP
80 PRINT "ESE CARACTER NO ES UN DIGITO"
90 END
```

Al principio pedimos el ingreso de un dígito. Luego comprobamos si lo es. Si es, lo imprimimos y cuando el programa encuentre el END LOOP saltará implícitamente a la línea 20 para seguir ejecutando las sentencias del lazo. Si no es dígito, el control se transfiere a la línea que continúa luego del END LOOP, imprimiéndose el mensaje correspondiente y finalizando la corrida del programa.



Sentencia: RCOMP
Formato: RCOMP: sentencia por verdadero: ELSE: sentencia por falso.
Función: Vuelve a ejecutar la última condición de una sentencia IF-THEN-ELSE.

SIMONS' BASIC

Esta sentencia sirve para no repetir las condiciones de una sentencia IF-THEN-ELSE. Analizando el siguiente ejemplo:

```
10 INPUT "INGRESE UNA LETRA";A$
20 IF A$="A" THEN PRINT "LETRA A": ELSE: PRINT "NO ES LA A"
30 RCOMP: PRINT "PRIMERA DEL ABECEDARIO": ELSE: PRINT "ES OTRA LETRA"
40 RCOMP: PRINT "CODIGO ASCII=65": ELSE: PRINT "CODIGO ASCII=";ASC(A$)
```

Si la letra ingresada es A se imprimirá: LETRA A PRIMERA DEL ABECEDARIO CODIGO ASCII=65. Si se ingresa cualquier otra: NO ES LA A ES OTRA LETRA CODIGO ASCII=" código correspondiente.

PROCEDIMIENTOS

Sentencia: CALL
Formato: CALL nombre del procedimiento o subrutina.
Función: Transfiere el control del programa al procedimiento identificado por su nombre.

Esta sentencia es similar al GOTO xxxx. Aquí el número de línea no se utiliza ya que a cada procedimiento (PROC) se lo identifica con un nombre. De esta manera cada vez que debamos utilizar un PROC lo invocaremos, simplemente, con su nombre sin interesarnos en qué número de línea se encuentre la primera sentencia de él. El ejemplo lo haremos luego de explicar la sentencia END PROC.

Sentencia: PROC
Formato: PROC nombre
Función: Identifica un procedimiento
La sentencia PROC se utiliza para rotular una subrutina o procedimiento.

Sentencia: END PROC
Formato: END PROC
Función: Indica el final del procedimiento o subrutina. Causa el mismo efecto que el RETURN del basic normal. El control retorna a la línea siguiente de la sentencia EXEC correspondiente. Sea el siguiente ejemplo:

```
10 INPUT "INGRESE UN NOMBRE CUYA LONGITUD NO SUPERE LOS 30 CARACTERES";A$
20 EXEC CHEQUEO
30 IFF=1THEN10
40 END
100 PROC CHEQUEO
110 F=0
120 IFLEN(A$))30THENF=1
130 END PROC
```

Para el caso de la sentencia CALL:

```
10 INPUT "INGRESE NUMERO";N%
20 IFN%=1THENCALL INGRESOS
30 CALL EGRESOS
40 END
50 PROC INGRESOS
60 (continúa el PGM)
200 PROC EGRESOS
210 (continúa el PGM)
```

Sentencia: EXEC
Formato: EXEC nombre del procedimiento
Función: Invoca al procedimiento correspondiente.
Esta sentencia es similar al GOSUB xxxx. La diferencia es igual que con CALL: no hace falta número de línea (xxxx)
El control se transfiere a la línea siguiente a EXEC cuando finalice el procedimiento.

TRUCOS

Control de sentencias Data

Con las sentencias que aquí les ofrecemos podrán controlar durante la ejecución de la sentencia READ el valor que se está leyendo juntamente con el número de línea donde se encuentra la sentencia DATA correspondiente. De esta forma podremos controlar la ejecución de ésta evitando, así, posibles errores de lectura.

```
100 READA:PRINTA,PEEK(63)+
    256*PEEK(64)
```

De esta manera cada vez que leemos imprimimos el valor con el número de línea donde éste se encuentra. Esto es muy útil cuando efectuamos programas en lenguaje de máquina y no disponemos de algún editor de assembler. Así podremos chequear fácilmente si el valor recientemente leído es mayor a 255 o menor a 0 (en ambos casos el intérprete imprimirá el mensaje de ILLEGAL QUANTITY ERROR).

Evitando el ? a través del GET

Como todos saben, trabajar con la sentencia INPUT implica que se imprime el signo de interrogación que, a veces, es indeseable. En notas anteriores les hemos dicho cómo pueden eliminar el ? de esta sentencia. Aquí les ofrecemos otra forma de ingresar datos por teclado, ahora utilizando la sentencia GET. Esta toma un sólo caracter a la vez y lo asigna a la variable en cuestión. Una forma de realizar esto es efectuando:

```
10 GETA$:IFA$=""THEN10
20 PRINTA ;:GOTO20
```

Prueben este método y comprobarán que el cursor no se imprime como en la entrada normal. Esto es muy incómodo ya que no sabemos cuál es la posición actual del cursor. Existe, sin embargo, otra forma de efectuar ésto. La forma es:

```
10 POKE 204,0
20 GETA$:POKE207,0:IFA$=""
   THEN20
30 POKE204,1:PRINTA$
```

Aquí el cursor se "prende" y se "apaga" normalmente. Finalmente también se puede utilizar:

```
5 B$=""
10 POKE207,0:POKE204,0:GETA
   $:IFA$=""THEN10
20 IFA$=CHR$(13)THENPRINT
   CHR$(32):GOTO40
30 PRINTA ;:B$=B$+A$:GOTO10
40 REM LA ENTRADA ESTA
   ALMACENADA EN B$
50 continúa el programa
```

LIST retardado

Si efectúan:
POKE56324,28:POKE56325,0
al ejecutarse el comando LIST éste se imprimirá más lento de lo normal. Para establecer el funcionamiento normal de la Drean Commodore 64 sólo deben oprimir las teclas de STOP y RESTORE simultáneamente.

Grabando en la C-16

Para aquellos que tienen la Drean Commodore 16 les sugerimos que cuando escriban un programa pongan en la primera línea:
1 REM Nombre del programa
De esta manera cada vez que ustedes deseen grabar el programa sólo se deben posicionar el cursor en la línea 1 y luego oprimir la tecla de función F5. Esta corresponde a DSAVE. Este se escribirá sobre la sentencia REM y el espacio. A continuación debemos oprimir RETURN para que la grabación se lleve a cabo.

MEZCLADOR DE PALABRAS

Tenemos que averiguar la posición correcta de un caracter en una frase dada (distinta) de la que ingresamos.

INGRESA LA PALABRA O FRASE QUE SERA MEZCLADA (MAXIMO 30 CARACTERES)

ESTA FRASE ES MIA

Con este entretenido juego debemos adivinar la posición correcta que un determinado caracter tiene en una palabra o frase dada. Claro que ésta no es igual a la que ingresamos: está totalmente mezclada; es decir que los caracteres que la constituyen tienen, ahora, una posición distinta, dentro de ella, con respecto a la original.
Al comienzo, el programa nos pide que ingresemos la cantidad de jugadores. Se debe ingresar un número comprendido entre 1 y 10. Si no se ingresa ningún valor, es decir se oprime la tecla RETURN, se asume 1. Cualquier otro valor no será aceptado y se interrogará nuevamente. Luego el programa pide la cantidad de palabras o frases a mezclar. El valor ingresado debe estar comprendido entre 1 y 10. A continuación se nos preguntará por los nombres de los participantes que intervienen en el juego. Una vez hecho esto la pantalla se borrará y, cada jugador, debe ingresar la palabra que él seleccionó para que algún otro la adivine. Esta debe tener una longitud menor o igual a los 30 caracteres y mayor o igual a 1 caracter. Este procedimiento se repite para cada uno de los jugadores. Cuando se halla completado esta tarea, el programa comenzará a mezclar las palabras. Luego comenzará el juego en sí. El turno de cada participante se imprimirá en la pantalla junto con la palabra original mezclada. Debajo de ella se encuentra otro casillero con redondeles los cuales representan las posiciones correctas de los caracteres. El jugador correspondiente debe adivinar el caracter correcto para la posición que el programa solicita. Si se ingresa un caracter erróneo, se imprimirá un cartel mostrando los errores cometidos. Si se ingresa el correcto, éste aparecerá debajo de la palabra mezclada; luego se solicitará el caracter de la posición siguiente. Se continúa así hasta que se complete la palabra, continuando con el siguiente jugador. El puntaje respectivo estará en función del número de errores cometidos y de la longitud de la palabra adivinada.

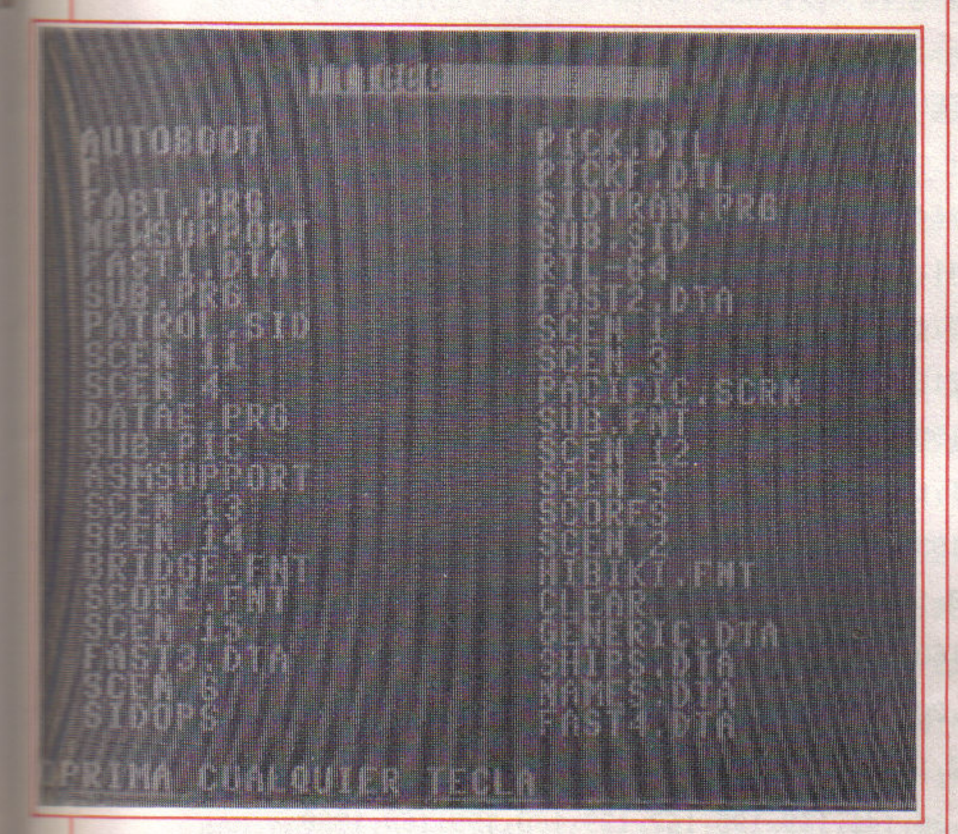
```
5 REM *** MEZCLADOR DE PALABRAS ***
10 SI=54272:KP=0
15 POKE53280,1:POKE53281,1:POKE646,6
20 DIMWL$(30),RI(30)
25 GOSUB455:INPUT"■■ CUANTOS JUGADORES ";PL:IFPL=0THENPL=1
28 IFPL>10THEN25
30 INPUT"■■ CUANTAS PALABRAS SE INGRESARAN ";NW:IFNW=0THENNW=1
32 IFNW>10THEN30
35 DIMWD$(PL-1,NW-1),WL(PL-1,NW-1),N$(PL-1),PS(PL-1)
40 FORK=0TOPL-1
45 GOSUB455:PRINT"■■ NOMBRE DEL JUGADOR ";K+1:INPUTN$(K)
50 FORX=0TONW-1
55 PRINT"■■■"N$(K)"!▒"
```

```
60 PRINT"▮ INGRESA LA PALABRA O FRASE QUE SERA      MEZCLADA";
65 PRINTTAB(08);" (MAXIMO 30 CARACTERES)▮▮"
70 PRINT"▮▮▮▮▮└";:FORJ=0TO29:PRINT"─";:NEXT:PRINT"┘▮"
75 PRINT"▮▮▮▮▮▮▮▮▮";:GOSUB365
80 L=LEN(W$):IFL>30THENGOSUB455:GOTO55
85 IFL=0THEN75
90 IFL=1THEN75
95 GOSUB465:GOSUB470:IFR=1THENGOSUB465:GOTO75
100 WD$(K,X)=W$:WL(K,X)=L:NEXTX:NEXTK
105 FORZ=0TONW-1:YY=0:FORY=PL-1TO0STEP-1:YY=YY+1:GOSUB455
110 W$=WD$(Y,Z):L=WL(Y,Z):SC=PS(YY-1):WS=L*200
115 PRINT"▮▮▮▮▮▮▮▮▮▮▮▮"TAB(14)"▮ MEZCLANDO▮▮"
120 FORJ=1TOL:WL$(J)=MID$(W$,J,1):NEXT
125 FORJ=1TOL:RI(J)=0:NEXT:GOSUB320:GOSUB345:FORJ=1TOL
130 I=INT(RND(0)*L)+1
135 FORC=1TOJ:IFRI(C)=ITHENC=J:NEXTC:GOTO130
140 NEXTC:RI(J)=I:NEXTJ
145 SW$="":FORJ=1TOL:SW$=SW$+WL$(RI(J)):NEXT:IFSW$=W$THEN125
150 GOSUB455:GOSUB485:C=(38-L)/2
155 PRINT"▮▮"TAB(C);:PRINT"▮|▮";SW$;"▮|▮"
160 PRINTTAB(C)"▮└";
165 FORJ=1TOL:PRINT"─";:NEXT:PRINT"┘▮"
170 PRINTTAB(C)"▮|▮";
175 FORJ=1TOL:PRINT"▮●▮";:NEXT:PRINT"▮|▮"
180 PRINTTAB(C)"▮└";
185 FORJ=1TOL:PRINT"─";:NEXT:PRINT"┘▮"
190 E=0:FORN=1TOL
195 PRINT"▮▮▮▮▮▮▮▮▮▮▮▮    LETRA PARA LA POSICION ";N;"▮"
200 GETL$:IFL$=""THEN200
205 PRINT"▮▮▮▮▮▮▮▮▮▮"TAB(19)"▮▮"L$"▮▮"
210 IFL$=WL$(N)THEN225
215 E=E+1:GOSUB320:GOSUB355:PRINT"▮▮▮▮▮▮▮▮▮▮▮▮▮▮▮ NUMEROS DE
    ERRORES=▮";E;"▮"
220 GOTO195
225 FORJ=1TOL:IFRI(J)=NTHENI=J:J=L:NEXT:GOTO235
230 NEXT
235 GOSUB320:GOSUB325
240 SL$=LEFT$(SW$,I-1):SR$=RIGHT$(SW$,L-I):SW$=SL$+"●"+SR$
245 PRINT"▮▮▮▮":PRINTTAB(C)"▮|▮";SW$;"▮|▮▮"
250 PRINT"▮▮▮▮▮▮▮";:PRINTTAB(C)"▮|▮"
```

```
255 FORJ=1TON:PRINTWL$(J);:NEXT:NEXT
260 PRINT"▒▒▒▒▒▒▒▒▒▒▒                              "
265 WS=WS-50*E:B=(L-E)*25:IFB<0THENB=0
270 SC=SC+WS+B:PRINT"▒▒▒PUNTAJE=▒";WS+B;"▒"
275 PS(YY-1)=SC:GOSUB490
280 PRINT"▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒"TAB(7)"OPRIMA RETURN PARA
   CONTINUAR"
285 GETA$:IFA$=""THEN285
290 IFA$<>CHR$(13)THEN285
295 GOSUB455:NEXTY,Z
300 GOSUB455:GOSUB340:GOSUB495
305 PRINT"▒▒▒JUEGAN NUEVAMENTE (S/N)";:INPUTA$
310 IFA$="S"THENCLR:GOTO10
315 PRINT"▒":END
320 GOSUB340:POKESI+24,15:RETURN
325 POKESI+5,12:POKESI+6,9
330 KP=KP*(KP<10)*(-1)+1:POKESI+1,KP*1.5+KP:POKESI+15,19+KP:
   POKESI+4,21
335 FORQ=1TO5:POKESI,Q*25:NEXT:RETURN
340 FORR=SITOSI+24:POKER,0:NEXT:RETURN
345 POKESI+5,50:FORX=1TO50:POKESI+1,RND(X)*32+50:POKESI+4,17
350 FORA=1TO10:NEXT:POKESI+4,16:NEXT:RETURN
355 POKESI+6,240:POKESI+4,17:FORA=1TO10:FORX=1TO255STEP25:POKESI-
   1,X:NEXT:NEXT
360 POKESI+4,32:RETURN
365 PRINT"▒▒▒▒▒";
370 W$="":CC%=0
375 GOSUB440:IFA$=CHR$(13)THENRETURN
380 IFA$=CHR$(34)THEN375
385 IFA$<CHR$(32)THEN375
390 IFA$>CHR$(127)THEN375
395 GOTO435
400 GOSUB440:IFA$=CHR$(13)THENRETURN
405 IFA$=CHR$(20)THEN425
410 IFA$<CHR$(32)ORA$>CHR$(127)THEN400
415 IFA$=CHR$(34)THEN400
420 GOTO435
425 CC%=CC%-1:IFCC%<0THEN370
430 W$=LEFT$(W$,CC%):PRINT"▒ ▒▒";:GOTO400
435 PRINTA$;:W$=W$+A$:CC%=CC%+1:GOTO400
440 PRINT"▒ ▒";:FORJ=0TO50:GETA$:IFA$<>""THENJ=50:NEXT:GOTO450
445 NEXT:PRINT"▒ ▒▒";:FORJ=0TO50:NEXT:GOTO440
450 PRINT"▒ ▒▒";:RETURN
455 PRINT"▒"TAB(12)"MEZCLADOR DE PALABRAS"
460 PRINTTAB(12)"▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒":RETURN
465 PRINT"▒▒▒▒▒▒▒▒▒▒▒                                   ";:RETURN
470 FORJ=1TOL:WL$(J)=MID$(W$,J,1):NEXT
475 R=0:FORJ=2TOL:IFWL$(1)<>WL$(J)THENJ=L:RETURN
480 NEXT:R=1:RETURN
485 PRINT"▒▒▒TURNO DE "N$(YY-1)
490 PRINT"▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒PUNTAJE:"
495 FORZZ=0TOPL-1:NN$=N$(ZZ):LL=LEN(NN$):FORXX=LLTO10:NN$=NN$+" "
   :NEXT
500 PRINT"▒"NN$"▒ ▒"PS(ZZ)"▒":NEXT
505 PRINT"▒▒":RETURN
```

DIRECTORIO SIMULTANEO

Este software muestra el directorio del diskette actual mientras continúa ejecutando el programa normalmente.



En ocasiones, cuando estamos trabajando con alguna de las bases de datos disponibles para la C-64 o algún otro utilitario, necesitamos saber el nombre de un determinado archivo.

Para ello debemos interrumpir el proceso actual y ejecutar el mando que imprime el directorio del diskette. Si se dispone del DOS 5.1, éste es el C$.

El programa que aquí les ofrecemos muestra el directorio del diskette actual mientras continúa ejecutando el programa normalmente. Para poder activarlo, primero deben cargar y ejecutar el listado que se encuentra más adelante. Luego cuando opriman la tecla de función 3 (F3) el directorio se imprimirá en la pantalla en dos columnas mostrando los archivos que constituyen el mismo sin los prefijos que indican su tipo (caso PRG, SEQ, REL, etc.). Para retornar a la pantalla anterior basta con presionar cualquier tecla. Desde ya, el directorio se imprimirá y, simultáneamente, continuará ejecutando el programa actual. Presionando las teclas RUN/STOP y RESTORE desactivarán este utilitario. Para activarlo se debe tipear SYS 49408. También es posible cambiar la tecla de función antes mencionada efectuando el POKE correspondiente en la dirección decimal 197. El tiempo de ejecución es muy rápido y puede trabajar juntamente con el DOS. Está escrito en lenguaje de máquina y ocupa desde la dirección $C100 hasta la $C3DA.

COMO TRABAJA

Primero se cambia el vector de dirección de salto de IRQ para que el programa pueda chequear, 60 veces por segundo, si se oprime la tecla F3. Si ésta fue presionada, las primeras cuatros páginas de memoria, es decir los primeros cuatro segmentos de 256 bytes, son copiadas bajo la ROM Basic (dirección decimal 40960). De ésta manera la memoria del programa no es destruida. Luego se comienza a leer el directorio del diskette y se convierten los códigos ASCII en códigos de Pokes para ponerlos en la memoria RAM. La ubicación de la pantalla en memoria es cambiada utilizando la técnica de cambio de pantalla para que las posiciones del directorio y el directorio sean impresos automáticamente (hay determinadas direcciones de memoria que indican el comienzo y fin del área de pantalla). Ahora, cuando nosotros presionemos cualquier otra tecla, la pantalla volverá a su estado anterior. (La ubicamos en las posiciones normales. Imaginen que la pantalla es una ventana que puede colocarse en cualquier parte de la memoria). Finalmente salimos de la rutina del directorio y reestablecemos la memoria.

COMO SE CARGA

Como habrán notado, el listado está formado básicamente por sentencias DATA. En caso de que ustedes se equivoquen en tipear los valores que acompañan a éstas, el programa les avisará del error cometido y, además, en qué número de línea fue (donde están los DATA). En este caso deben listar la línea correspondiente y corregir el valor erróneo.

El chequeo antes citado se realiza de la siguiente manera: cada sentencia DATA está formada por 13 valores numéricos. Los primeros 12 son las instrucciones en decimal del código de máquina. El valor numérico 13 es la suma real de los anteriores. Al leerlos se efectúa una sumatoria de esos valores y se comprueba que sea igual al valor del décimo tercer elemento numérico.

Simultáneamente se lleva un contador de línea. De esta manera si la sumatoria es distinta al valor real, el programa informa del error imprimiendo el valor actual del contador de línea. Caso contrario sigue cargando normalmente.

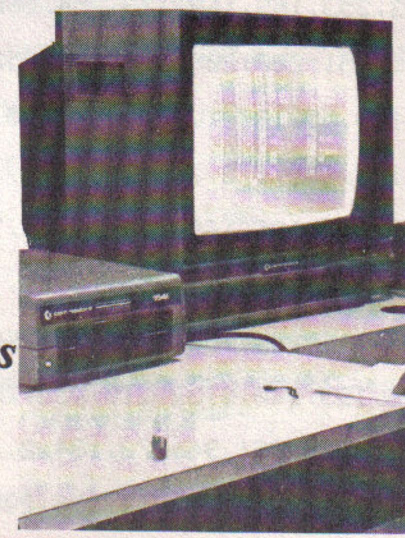
PROGRAMAS

```
1 PRINT"J":DI=49408:PRINTTAB(10);"■■■■■UN MOMENTO POR
FAVOR":LIN=10
2 CO=DI:LOT=55:FORI=1TOLOT:SUM%=0
3 FORJ=1TO12:READNUM%:IFNUM%>255THEN8
4 POKEDI+J-1,NUM%:SUM%=SUM%+NUM%:NEXTJ
5 READVA%:IFVA%<>SUM%THEN8
6 DI=DI+12:LIN=LIN+10:NEXTI:PRINT"■■"
7 PRINTTAB(10);"DIRECTORIO ACTIVADO":PRINTTAB(10)"OPRIMA
F3":SYSCO:NEW
8 PRINT"ERROR EN LA SENTENCIA DATA NUMERO ";LIN:STOP
9 REM *** DATAS ***
10 DATA 120,173,20,3,141,211,195,173,21,3,141,212,1413
20 DATA 195,169,33,141,20,3,169,193,141,21,3,169,1257
30 DATA 0,141,213,195,141,216,195,88,96,165,197,201,1848
40 DATA 5,208,5,173,213,195,240,3,108,211,195,169,1725
50 DATA 255,141,213,195,56,32,240,255,142,214,195,140,2078
60 DATA 215,195,169,1,133,204,165,251,141,207,195,165,2041
70 DATA 252,141,208,195,165,253,141,209,195,165,254,141,2319
80 DATA 210,195,169,0,133,251,169,8,133,252,169,0,1689
90 DATA 133,253,169,160,133,254,160,0,162,4,177,251,1856
100 DATA 145,253,200,208,249,230,252,230,254,202,208,242,2673
110 DATA 32,90,195,32,135,194,169,52,133,251,169,8,1460
120 DATA 133,252,169,36,141,139,195,162,139,160,195,169,1890
130 DATA 1,32,189,255,169,14,160,96,162,8,32,186,1304
140 DATA 255,32,192,255,169,8,32,180,255,169,96,32,1675
150 DATA 150,255,230,199,160,0,132,144,32,165,255,166,1888
160 DATA 144,208,74,201,34,208,241,32,165,255,166,144,1872
170 DATA 208,63,201,34,240,5,32,52,195,208,240,165,1643
180 DATA 199,240,8,198,199,169,122,133,251,208,213,24,1964
190 DATA 169,20,101,251,133,251,144,2,230,252,165,251,1969
200 DATA 201,154,208,196,165,252,201,11,208,190,32,234,2052
210 DATA 194,32,108,194,32,90,195,169,42,133,251,169,1609
220 DATA 8,133,252,208,171,169,14,32,195,255,32,171,1640
```

```
230 DATA 255,32,161,194,32,135,194,32,234,194,32,108,1603
240 DATA 194,169,54,133,1,169,0,133,251,169,8,133,1414
250 DATA 252,169,0,133,253,169,160,133,254,160,0,162,1845
260 DATA 4,177,253,145,251,200,208,249,230,252,230,254,2453
270 DATA 202,208,242,169,55,133,1,173,207,195,133,251,1969
280 DATA 173,208,195,133,252,173,209,195,133,253,173,210,2307
290 DATA 195,133,254,32,15,195,169,0,133,204,174,214,1718
300 DATA 195,172,215,195,24,32,240,255,169,0,141,213,1851
310 DATA 195,76,188,254,162,24,160,0,24,32,240,255,1610
320 DATA 160,0,185,115,195,240,6,32,210,255,200,208,1806
330 DATA 245,32,228,255,240,251,96,173,134,2,160,0,1816
340 DATA 153,0,216,153,0,217,153,0,218,136,208,244,1698
350 DATA 160,232,153,255,218,136,208,250,96,169,8,32,1917
360 DATA 180,255,169,111,32,150,255,160,0,32,165,255,1764
370 DATA 153,139,195,200,192,40,240,4,201,13,208,241,1826
380 DATA 169,0,153,139,195,160,0,185,139,195,201,50,1586
390 DATA 144,28,173,216,195,208,3,32,234,194,160,0,1587
400 DATA 162,0,24,32,240,255,185,139,195,32,210,255,1729
410 DATA 200,185,139,195,208,247,32,171,255,96,173,216,2117
420 DATA 195,208,31,169,255,141,216,195,160,25,185,217,1997
430 DATA 0,153,180,195,136,16,247,173,24,208,41,15,1388
440 DATA 9,32,141,24,208,169,8,141,136,2,96,173,1139
450 DATA 216,195,240,31,169,0,141,216,195,160,25,185,1773
460 DATA 180,195,153,217,0,136,16,247,173,24,208,41,1590
470 DATA 15,9,16,141,24,208,169,4,141,136,2,96,961
480 DATA 41,127,201,64,144,10,201,97,176,4,41,191,1297
490 DATA 144,2,41,223,141,217,195,165,199,240,8,173,1748
500 DATA 217,195,9,128,141,217,195,173,217,195,145,251,2083
510 DATA 200,96,169,32,160,0,153,0,8,153,0,9,980
520 DATA 153,0,10,136,208,244,160,232,153,255,10,136,1697
530 DATA 208,250,96,79,80,82,73,77,65,32,67,85,1194
540 DATA 65,76,81,85,73,69,82,32,84,69,67,76,859
550 DATA 65,0,0,0,0,0,0,0,0,0,0,65
```

ASSEMBLER

Hay instrucciones que nos permiten efectuar incrementos y decrecimientos en los contenidos de algunos registros en forma explícita.



En el número anterior comenzamos a introducirnos dentro del lenguaje máquina. Comentamos que, junto con éste, iríamos describiendo el assembler correspondiente. Creemos que el entendimiento de este último será más fácil si asimilamos, antes que nada, el primero. La experiencia así lo demuestra. De todas maneras les sugerimos que profundicen sobre el tema. Existe en la actualidad una gran variedad de textos que describen con más profundidad el tema aquí expuesto.

Incrementos-decrecimientos explícitos:

Hay instrucciones que nos permiten incrementar los contenidos de algunos registros en forma explícita. Sí, por ejemplo, el caracter T representa al registro X, Y o a la memoria (m), la operación que realiza esta instrucción es:

(T)+1 —> T

Es decir que al contenido del registro representado por T se le suma 1 y el resultado es almacenado nuevamente en T. Esta instrucción requiere de 1 Byte solamente; su código de operación (CO). Por ejemplo el CO para incrementar el contenido del registro X es $E8. Para el registro Y y para la memoria es $C8 y $E6 respectivamente.

Para el caso de los decrecimientos ocurre lo mismo con la salvedad de que la operación que se efectúa es:

(T)-1 —> T

Los CO para decrecer los registros X, Y, M son $CA, $88, $C6.

Este último lo podemos realizar de cuatro maneras (también vale para incrementos de M solamente). Estos son:

1) PAGINA CERO
2) PAGINA CERO, X
3) ABSOLUTA
4) ABSOLUTA, X

Desde ya éstos tienen que ver con los modos de direccionamientos explicados en la nota anterior. El primero requiere de 2 Bytes CO y parte baja de la dirección (recuerden que la parte alta es siempre cero dentro de este rango de memoria).

El segundo se refiere al direccionamiento indexado página cero. Aquí el registro que interviene es el X.

El tercero es direccionamiento absoluto (CO, parte baja, parte alta).

Finalmente el último es el direccionamiento indexado.

Por supuesto todos ellos tienen como finalidad determinar la dirección efectiva (DE) para luego realizar:

(M)+1 —> M o (M)-1 —> M

según sea el caso.

Para completar los conceptos, realicemos tres ejemplos de incrementos para cada uno de los registros. Para ello utilicemos el editor de absoluto publicado en el número 1 y cuyo nombre es ASSEMBLER 1.1:

```
DIRECCION INICIAL: ? 3000
3000 ? A2 (CO carga reg.X en forma
          inmediata con $AA)
3001 ? AA
3002 ? E8 (inc. reg X)
3003 ? 60 (RTS)
3004 ? .
       ? R (ejecutamos el PGM)
3004 ? .
       ? X (imprime contenido reg X)
     AB
```

Para hacer el incremento del registro Y debemos cambiar el contenido de la dirección $3000 por $A0 (CO carga reg Y en forma inmediata) y de la $3002 por $C8 (incrementa reg Y).

Para la memoria haremos uno de los casos: Absoluto. El resto los dejamos para que los realicen ustedes. La comprobación es muy sencilla; simplemente hay que verificar que el contenido de la dirección dada esté incrementada en uno luego de la ejecución del CO. Estos son $E6, $F6, $FE para los direccionamientos 1, 2 y 4 citados anteriormente.

```
DIRECCION INICIAL : ? 3000
3000 ? FE (ponemos el valor a
       incrementar en la dirección
       elegida)
3001 ? .
       N (seteamos el editor. El valor
       en $ 3000 seguirá siendo $FE)
DIRECCION INICIAL : ? 4AAA
4AAA ? EE (CO inc memoria absoluta
       con la dirección $3000)
4AAB ? 00 (byte bajo de esa dirección)
4AAC ? 30 (byte alto)
4AAD ? 60 (RTS)
4AAE ? .
       :? R
4AAE ? .
       ?L3000 (listamos el contenido
       de $3000)
3000   FF
```

Luego el contenido de la dirección $3000 fue incrementado en uno.

Saltos relativos:

La forma para transferir el control del programa (PGM) dentro de él varía de acuerdo con el microprocesador que se utilice. Hay algunos en que se indica la dirección de salto en forma explícita pudiéndose, de esta manera, saltar a cualquier dirección. En otros se debe calcular la dirección de salto. Este indica el valor que el micro debe sumar al PC actual para acceder a esa dirección determinada.

A ese valor se lo conoce con el nombre de OFFSET .

Este es el caso del micro de la C-64 (6510; también del 6502 y del 6800). Con este método se puede saltar, a lo sumo, a la dirección 127 "delante" del PC (es decir a la dirección representada por (PC)+$7F lo que sería saltos hacia adelante) y a la dirección 128 "detrás" del PC (es decir a la dirección representada por (PC)+$80 lo que sería saltos hacia atrás.



Coloquen en la dirección $5F1A el valor de $FF. Luego escriban el programa correspondiente para cargar el registro Y con el contenido de aquella dirección (es decir en forma absoluta; CO=$AC) e incrementen en uno el contenido de este registro.
El programa se deberá cargar a partir de la dirección $ 3000. No olviden finalizar el programa con RTS (CO= $60). Ejecuten éste y observen el contenido del registro Y.
Para desarrollar esta tarea les sugerimos que utilicen el programa ASEMBLER 1.1, cuyo listado fue publicado en el número 1 de esta revista.

Lo dicho se refiere a saltos condicionales, por los cuales el control de programa se transferirá si se satisface una determinada condición. Sin embargo es posible transferir en control independientemente de condición alguna. Esto se conoce con el nombre de salto incondicional.
Efectuando una analogía con el Basic, podemos decir que lo primero corresponde a la sentencia IF condition THEN GOTOxxxx y la segunda al GOTOxxxx.

SALTOS INCONDICIONALES:
En el caso de la C-64, los saltos incondicionales se efectúan a través de la instrucción JMP (JUMP-salto).
Existen dos tipos:
1) saltos absolutos (CO=$4C):
La dirección de salto está a continuación del CO en formato byte alto, byte bajo. Como habrán notado, esta instrucción requiere de tres bytes (CO,byte alto de la dirección, byte bajo). Un ejemplo de ello sería:

3001 4C
3002 D2
3003 FF

De esta forma saltaríamos a la dirección $FFD2.
2) saltos indirectos (CO=$6C):
Aquí también se requieren de tres bytes de la misma manera que en el caso anterior. La diferencia radica en que el salto se efectúa al contenido de la dirección que sigue a continuación del CO. Analicemos el siguiente ejemplo:

3002 6C
3003 00
3004 30
3000 00
3001 40
4000 R9
4000 A9
4001 AF
4002 60

En las direcciones 3002, 3003 y 3004 efectuamos el salto al contenido de la dirección que allí se encuentra; es decir saltamos a ($3000). El micro sabe que lo primero con que se encuentra, en esta dirección, es el byte bajo de la dirección "final" de salto y que a continuación se encuentra con el byte alto de dicha dirección. En nuestro ejemplo esos bytes se encuentran respectivamente en la dirección $3000 y $3001 ($00 y $40). Finalmente se toman éstos si se completa el salto transfiriendo el control a la dirección $4000 donde se carga el acumulador con $AF y se para la ejecución del programa. Utilizando el assembler 1.1 traten de practicar ambos saltos. Recuerden que, en el caso del salto indirecto, se debe ejecutar el programa a partir de la dirección donde se encuentra el CO de la instrucción (en nuestro caso a partir de la dirección $3002).

Registro del estado del microprocesador.

Antes de continuar explicando los saltos en la C-64, debemos comentar el registro de estado del microprocesador o comunmente llamado registro P. Todos los saltos condicionales, que más adelante explicaremos, utilizan alguno de los bits de este registro. Las condiciones que deben satisfacerse para que el salto se lleve a cabo son, justamente el "seteo" o "reseteo" de los valores de dichos bits. Este registro, como su nombre lo indica, representa el estado actuál del microprocesador. Esta constituido por siete flags:

bit Nro.:	7	6	5	4	3	2	1	0
	N	V	-	B	D	I	Z	C

El significado de cada uno de estos son:
C (Carry): Este flag se pone en "1" (se setea) si hubo acarrero proveniente del bit 7. Este bit también se modifica como resultado de operaciones específicas o por las instrucciones SEC o CLC (SEt Carry-setear carry - o Clear Carry-borrar carry).
Se utiliza como un noveno bit cuando se ejecutan instrucciones de rotación y de shift (luego las explicaremos).
Las instrucciones que afectan a este flag son: ADC, ASL, CLC, CMP, CPX, CPY, LSR, POP, ROL, ROR, RTI, SBC y SEC.
Z (Zero): Indica si el resultado de una operación artimética fue cero con el valor lógico "1" o con "0" si no lo fue.
I (interrup): Flag de interrupción. Este es un bit muy importante en la programación del microprocesador. Si este bit está en "1", se "enmascara" la línea de requerimiento de interrupción (IRQ). De esta manera si ocurre una interrupción cuando este bit esté en "1", la C-64 no la atenderá.
D (decimal mode): Indica modo decimal. Este puede ser seteado o reseteado por las instrucciones SED (SEt Decimal mode) y CLD (Clear Decimal mode). Si este bit está en "1", produce que todas las operaciones de suma tengan un ajuste decimal.
B (break): Flag de break (BRK). Este es usado internamente por el microprocesador para indicar si el ingreso de una rutina de interrupción fue ocasionada por la ejecución de la instrucción BReaK o por un requerimiento de interrupción externo. No existen instrucciones que puedan setear o resetear este bit.
-: no se utiliza
V (overflow): Flag de Overflow. Se pone a "1" si se exede la capacidad del acumulador para almacenar números en complemento a dos. Las operaciones que pueden afectar a este bit son las de sumas, restas, BIT, CLV, PLR, RTI.
N (negative): Indica si el resultado de una operación fue negativo con un "1" o con un "0" si fue positivo. Este bit no puede ser seteado o reseteado por el programador.

Es decir que si nuestra condición de salto es por menor que cero, ésta se llevará a cabo si el flag N está en "1"; o si la condición es saltar si es cero, ésta se llevara a cabo si el flag Z está en "1", etc. Si observan el set de instrucciones (lo publicaremos en el próximo número), verán que para cada una de las instrucciones de salto figura, al lado de éssta, con que flag bifurca y cuál debe ser su valor para que ello se cumpla.

Volviendo a los saltos relativos.

Estamos ahora en condiciones de encarar estos saltos. En la nota anterior habíamos dicho que es necesario determinar una constante u OFFSET la cual, sumada al contenido actúal del PC, "apunta" a la dirección de salto. También habíamos dicho que podíamos realizar saltos a la posición 127 delante del PC y 128 atras de él; es decir valores del offset de $7F y $80 respectivamente. No olviden que $80 es un número negativo. De todas maneras analicen al tabla 1

Como habran notado el bit 7 representa el signo del número. Esa tabla explica porque no se pueden efectuar saltos más extensos. Recuerden que restar un número es lo mismo que sumar su inverso aditivo. Esto lo decimos por la operación aritmética que realiza el micro cuando debe acceder a una nueva dirección. para explicar este cálculo pongámonos de acuerdo en lo siguiente: cuando se mencione el contenido actual de contado de programa (PC) se debe entender que éste apunta a la dirección donde se encuentra la instrucción de salto. El conjunto de ésta con el offset se representa de la siguiente manera:
dddd instrucción de salto
dddd+1 offset
donde dddd es una determina dirección de memoria. En este caso el contenido actúal del PC, de acuerdo a nuestra convención, es dddd.
Finalmente la dirección de salto se calcula de la siguiente manera:
dirección de salto = (PC) + 2 = K -1-
donde K representa al offset (en hexadecimal). Realicemos una serie de ejemplos para que quede un poco más claro:

dirección	instrucción
4000	FO inst.
4001	06 offset
4002	xx
4003	xx
4004	xx
4005	xx
4006	xx
4007	xx
4008	A9
4009	DD
400R	60

Tabla 1

	binario							decimal
Bit Nro: 7	6	5	4	3	2	1	0	
0	1	1	1	1	1	1	1	127
0	1	1	1	1	1	1	0	126
0	1	1	1	1	1	0	1	125
0	1	1	1	1	1	0	0	124
....								
0	0	0	0	0	0	1	1	3
0	0	0	0	0	0	1	0	2
0	0	0	0	0	0	0	1	1
0	0	0	0	0	0	0	0	0
1	1	1	1	1	1	1	1	-1
1	1	1	1	1	1	1	0	-2
1	1	1	1	1	1	0	1	-3
1	1	1	1	1	1	0	0	-4
....								
1	0	0	0	0	0	0	1	-127
1	0	0	0	0	0	0	0	-128

La primera instrucción que aparece es BEQ (Branch on Equal Zero-Saltar si es cero) la cual significá saltar si el flag Z es "1". En caso en que ésto se cumpla, cuando el PC Llegue a la dirección $4000, el control se transferirá a la dirección dada por la ecuación -1-, es decir:
dirección de salto = (PC) + 2 + K = $4000 + $2 + $6 = $4008
Aquí se carga el acumulador con $DD y finaliza la ejecución del programma. (las xx significan otras instrucciones)

dirección	instrucción
$4000	70 ins.
$4001	0A offset
$4002	xx
$4003	xx
$4004	xx
......	....
$400C	A0
$400D	A1
$400E	60

CALCULOS MATEMATICOS

Software para realizar operaciones como el factorial de un número, el máximo común divisor entre dos números, o la derivada de una función en un punto dado.

```
FUNCION A UTILIZAR

POR FAVOR, COLOQUE LA FUNCION SELECCIO
NADA INGRESANDO :

5 DEF FNY(X)= FUNCION

DESEA SEGUIR (S/N)
```

Este programa les permitirá efectuar una serie de cálculos matemáticos como ser factorial de un número, máximo común divisor entre dos números, derivada de una función en un punto dado, integral definida de una función y, finalmente, ceros (raíces) de una función. En los últimos tres casos se debe ingresar la función en la línea 5. Si se quiere modificar la actual, sólo debemos hacer:

5 DEF FNY (X) = función deseada

Cuando corramos el programa aparecerá el menú correspondiente identificando a cada procedimiento con un número. Tendremos de esta forma que el 1 es RAICES, 2 FACTORIAL, 3 M C D, 4 DERIVADA, 5 INTEGRAL, 6 FIN. Luego se nos interrogará sobre el número de opción seleccionado. Debemos ingresar un valor comprendido entre 1 y 6; caso contrario se nos interrogara nuevamente.

Para las opciones correspondientes a factorial y máximo común divisor se debe proceder de acuerdo a las propias instrucciones de esas rutinas. En el primer caso, el número que ingresaremos para hallar el factorial de él debe estar comprendido, por razones de almacenamiento interno de la C-64, entre 0 y 33; de otra forma se nos indicará del error cometido a través del correspondiente mensaje. Además si el número es menor que cero (negativo) se calculará el factorial de su valor absoluto.

Para el MCD (máximo común divisor) sólo debemos ingresar los dos números y la rutina nos informará rápidamente del resultado obtenido.

Si seleccionamos la opción 1, el programa nos pedirá los extremos donde se encuentra la raíz buscada. Estos no pueden ser cualquiera: el signo de la función, en estos puntos, deben ser distintos. Así se asegura que entre éstos exista un cero de la función. Por ello el algoritmo no puede calcular raíces complejas. Otro de los valores que se piden aquí es el número de pasos. Estos representan el número de particiones que hará éste hasta llegar a la raíz de la función. A mayor números de pasos mayor será la aproximación de la raíz de la función.

Al seleccionar la opción 4 se nos interrogará sobre en número de pasos. Estos representan lo mismo que el caso anterior.

Luego se nos pedirá el punto de tangencia, es decir el punto donde hallaremos la pendiente de la recta

tangente a la función dada.
En la opción 5 se nos pedirá el número de pasos pero, aqui, no se verifica,. como la teoria lo demuestra, que a mayor número de pasos mayor aproximación del área de la función (integral definida). Puede suceder que aumente la aproximación de la integral o que aumente el error de cálculo. Esto ocurre por un problema de redondeo y truncamiento de los valores internamente almacenados. Confirmen ustedes mismos estos últimos ingresando distintos valores de pasos. A continuación se nos interrogará sobre los límites de integración (inferior, superior). No podremos ingresar el mismo límite inferior y superior es decir 1,1; 21.3, 21.3; etc. Esta rutina utiliza el método de aproximación de los trapecios.

```
1 REM CALCULOS MATEMATICOS-CRISTIAN J PARODI
5 DEF FNY(X)=X↑2-4
10 PRINT"▒▒"
11 PRINT"▒▒▒▒          ┌────────────────────┐"
15 PRINTTAB(7);"       │CALCULOS MATEMATICOS │"
20 PRINTTAB(7);"       └────────────────────┘"
22 PRINT"▒▒              1.............RAICES"
25 PRINT"▒              2.............FACTORIAL"
30 PRINT"▒              3.............M C D"
35 PRINT"▒              4.............DERIVADA"
40 PRINT"▒              5.............INTEGRAL"
42 PRINT"▒              6.............FIN"
45 POKE19,64:INPUT"▒▒▒▒                    SU OPCION :";OP%:PRINT
   :POKE19,0
50 IFOP%>=1ANDOP%<=6THEN55
52 GOTO10
55 PRINT"▒"
60 ONOP%GOSUB200,100,300,500,700,80
70 GOTO10
80 END
99 REM *** FACTORIAL ***
100 F=4:C=14:TEX$="FACTORIAL":GOSUB900
110 POKE19,64:INPUT"▒▒▒▒▒▒  INGRESE NUMERO  :";NU%:PRINT:POKE19,0
111 NU%=ABS(NU%):IFNU%>33THEN160
115 FAC=1:FORI=1TONU%
120 FAC=FAC*I
125 NEXT
130 PRINTTAB(2)"▒▒EL FACTORIAL DE ";ABS(NU%);" ES ";FAC
140 GOSUB990:IFFL=1THENPRINT"▒":GOTO100
150 RETURN
160 PRINT"▒EL FACTORIAL DE ";ABS(NU%);" SUPERA LA MAGNITUD DE
    ALMACENAMIENTO";
165 PRINT" INTERNO DE LA MAQUINA"
170 GOSUB990:IFFL=1THENPRINT"▒":GOTO100
175 RETURN
199 REM *** RAICES DE UNA FUNCION ***
200 GOSUB1000
202 PRINT"▒"
210 F=3:C=10:TEX$="RAICES DE UNA FUNCION"
211 GOSUB900
215 PRINT"▒   INGRESE LOS EXTREMOS DONDE SE"
220 PRINT"   ENCUENTRA LA RAIZ"
221 PRINT"    RECUERDE QUE LOS SIGNOS DE LA FUNCION   EVALUADA";
222 PRINT" EN A Y B DEBEN SER DISTINTOS"
225 POKE19,64:INPUT"    A , B :";A,B
230 PRINT
235 POKE19,0
236 POKE19,64:INPUT"    NUMERO DE PASOS :";N%:IFN%<=0THEN202
```

```
237 PRINT:POKE19,0
238 L=SGN(FNY(A)):R=SGN(FNY(B))
240 IFL<>RTHEN243
241 GOTO202
243 FORI=1TON%
245 M=(A+B)/2:FM=SGN(FNY(M))
248 IFFM<>RTHENA=M:GOTO255
250 B=M
255 NEXTI
260 PRINT"⊠⊠  LA RAIZ DE LA FUNCION ES"
262 PRINT"  APROXIMADAMENTE :";M
264 PRINT"⊠⊠  Y(";STR$(M);")=";FNY(M)
270 GOSUB990:IFFL=1THEN202
275 RETURN
299 REM *** M C D ***
300 F=4:C=17:TEX$="M C D":GOSUB900
310 POKE19,64:INPUT"⊠⊠⊠  INGRESE A,B :";A%,B%:PRINT:POKE19,0
315 A1%=A%:B1%=B%
320 Q%=A%/B%:R%=A%-(Q%*B%)
330 A%=B%
335 B%=R%
340 IFR%>0THEN320
345 PRINT"⊠⊠⊠ EL M C D ENTRE ";A1%;" Y ";B1%;" ES ";A%
350 GOSUB990:IFFL=1THENPRINT"⊐":GOTO300
355 RETURN
499 REM *** DERIVADA ***
500 GOSUB1000
502 PRINT"⊐"
505 F=5:C=10:TEX$="DERIVACION NUMERICA"
510 GOSUB900
515 POKE19,64:INPUT"⊠⊠⊠ NUMERO DE PASOS :";N%:PRINT:POKE19,0
517 IFN%<=0THEN502
520 POKE19,64:INPUT"⊠ PUNTO DE TANGENCIA (X0) :";X0:PRINT:POKE19,0
530 DE=(FNY(X0+1/N%)-FNY(X0))*N%
540 PRINT"⊠⊠ LA DERIVADA DE LA FUNCION EN EL PUNTO :":PRINT;
550 PRINTTAB(15);X0;",";FNY(X0):PRINT
555 PRINT" ES :";DE
560 GOSUB990:IFFL=1THEN502
565 RETURN
699 REM *** INTEGRACION NUMERICA ***
700 GOSUB1000
702 PRINT"⊐"
705 F=3:C=10:TEX$="INTEGRALES DEFINIDAS"
710 GOSUB900
715 POKE19,64:INPUT"⊠⊠⊠ NUMERO DE PASOS :";N%:PRINT:POKE19,0
720 IFN%<=0THEN702
725 POKE19,64:PRINT"⊠   LIMITE INFERIOR Y SUPERIOR "
728 PRINT"   DE INTEGRACION ":INPUT"   X0 , XN :";X0,XN:PRINT
    :POKE19,0
729 IFX0=XNTHEN702
730 H=ABS((XN-X0)/N%)
735 C=0:B1=FNY(X0)
740 FORI=X0TOXN-HSTEPH
745 B2=FNY(X0+(C+1)*H)
750 A=A+H*((B1+B2)/2):B1=B2:C=C+1
```

PROGRAMAS

```
755 NEXTI
760 PRINT"▮▮ LA INTEGRAL DE LA FUNCION ENTRE LOS      PUNTOS":PRINT;
765 PRINTTAB(15);X0;" Y ";XN:PRINT
770 PRINT" ES :";A
780 GOSUB990:IFFL=1THEN702
785 RETURN
899 REM *** SE IMPRIMEN LOS CARTELES ***
900 POKE214,F:POKE211,C:SYS58640:PRINTTEX$
910 RETURN
989 REM *** GET ***
990 FL=0:F=20:C=10:TEX$="DESEA SEGUIR (S/N)":GOSUB900
995 GETA$:IFA$=""THEN995
996 IFA$="S"THENFL=1:RETURN
997 IFA$="N"THENRETURN
998 GOTO995
999 REM *** SOBRE LA FUNCION ELEGIDA ***
1000 F=8:C=12:TEX$="FUNCION A UTILIZAR"
1005 GOSUB900
1010 PRINT"▮▮ POR FAVOR, COLOQUE LA FUNCION SELECCIO  NADA
     INGRESANDO : "
1015 PRINT"▮▮"
1020 PRINT"5 DEF FNY(X)= FUNCION
1030 F=16:C=15:TEX$="": GOSUB900
1035 GOSUB990:IFFL=1THENRETURN
1040 STOP
```

TRUCOS

Días de la semana

Con esta rutina podremos determinar el día de la semana correspondiente a una fecha comprendida entre el 1 de marzo de 1900 y el 28 de febrero del año 2000. Nosotros sólo debemos ingresar el mes, día y año de la fecha que queremos averiguar. Este último se especifica con sólo ingresar los últimos dos números (86,90,99, etc.). Otra de las cosas importantes de resaltar es que la variable DS$ que representa el vector con los días de la semana, no se destruye. Es por ello que podemos utilizar esta rutina todas las veces que querramos.

```
10 REM DIA DE SEMANA
20 DATADOM,LUN,MAR,MIE,
   JUE,VIE,SAB
30 FORJ=0TO6:READDS$(J):
   NEXT
40 INPUT"MES,DIA,AÑO";MM,
   DD,AA
50 CY=AA:M=MM-2:IFM<
   1THENM=M+12:CY=CY-1
60 Y=CY-INT(CY/100)*100
70 DS=Y+INT(Y/4)+1+DD+INT
   (2.6*M-.1999)
80 DS=DS-INT(DS/7)*7:PRINT
   DS$(DS)
```

Averigua SYS

Este pequeño programa les permitirá averiguar la dirección inicial de un determinado programa almacenado en diskett. El programa les entregará la dirección de arranque del programa juntamente con la parte alta y baja de esa dirección (ambas en decimal). De esta manera podrán realizar el SYS correspondiente. Este utilitario es muy útil cuando tenemos programas (juegos, otros utilitarios, etc.) de los cuales desconocemos la dirección inicial.

```
10 REM AVERIGUA SYS
20 INPUT"NOMBRE DEL
   PROGRAMA";NP$
30 OPEN2,8,2,NP$+",P,R"
40 GET#2,LO$:LO$=LO$+CHR
   $(0)
50 GET#2,HI$:HI$=HI$+CHR$(
   0)
60 CLOSE2:DI=ASC(LO$)+256*
   ASC(HI$)
70 PRINT"LA DIRECCION
   INICIAL ES:";DI
80 PRINT" BYTE ALTO:";HI$
90 PRINT" BYTE BAJO:";LO$
100 END
```

USANDO POSICIONES DE MEMORIA

En más de una ocasión debemos utilizar variables para indicar, con determinados valores, errores, ingresos, etc. o para tomar decisiones en nuestro programa. Es decir que gracias a estos "indicadores" o flags realizamos diversas tareas. Pero a medida que necesitamos más indicadores debemos aumentar la cantidad de variables con lo que se reduce la memoria libre del C-64. Existe una manera de tener todos los flags necesarios sin consumir memoria. Esto se logra utilizando las direcciones de memoria que no son usadas por el sistema operativo de la C-64. Un ejemplo de ello es la dirección $0002 Supongamos que nuestro programa necesita 8 flags los cuales indican, por ejemplo, si un número es mayor que 100, si es menor que 0, si es cero, si es mayor que 1000, si es uno, si es mayor que 10, si es entero y, finalmente, si es de punto flotante. Supongamos, también, que un "1" representa una afirmación y un "0" una negación. Si utilizamos el método convencional, necesitaríamos, como antes dijimos, ocho variables. Los nombres de éstas

deben ser, por una cuestión de mayor entendimiento, acordes a lo que ellas representan. Por ejemplo, ellas pueden ser MAY100, MEN0, CERO, MAY1000, UNO, MAY10, ENTERO y REAL. Así, si MAY100 es igual a 1, indicará que el número es mayor que 100 o si es 0 el número no es mayor a 100. Imaginen ustedes la incomodidad de trabajar con estas ocho variables.

La otra forma es utilizando la manera no convencional, es decir direcciones de memoria. Para el mismo ejemplo utilicemos la dirección $0002. Su contenido está formado por un byte, es decir 8 bits. Cada uno de ellos nos puede representar las ocho variables antes mencionadas. Sólo debemos ponernos de acuerdo qué bit representará los diversos flags. Por ejemplo podríamos hacer la siguiente asignación:

bit 7 (mayor peso) = mayor que 100
bit 6 = menor que 0
bit 5 = cero
bit 4 = mayor que 1000
bit 3 = uno
bit 2 = mayor que 10
bit 1 = entero
bit 0 (menor peso) = real (punto flotante)

Luego si alguno de estos bits es un "1" lógico, indicará que se está en el caso respectivo mientras que con un "0" será lo contrario. Por ejemplo el número 10 originará que el contenido de esta dirección sea:

```
Bit nro.: 7 6 5 4 3 2 1 0
          0 0 0 0 0 0 1 0
```

Con el mismo criterio los números 100023,0,109.3,-12 ocasionarán que el contenido de esta dirección sea:

Bit nro. número	7	6	5	4	3	2	1	0
100023	1	0	0	1	0	1	1	0
0	0	0	1	0	0	0	1	0
109.3	1	0	0	0	0	1	0	1
-12	0	1	0	0	0	0	1	0

A través de las sentencias PEEK, POKE, AND, podemos fácilmente representar y determinar la característica del número leído. Para saber si el número es, por ejemplo, entero, debemos realizar una AND entre el contenido de la dirección $0002 y 00000010 (2 decimal). Si, como resultado, obtenemos el número 2, ésto indicará que el número es entero. Es decir que PEEK(2)AND2 debe ser igual a 2. Igualmente debemos cambiar los bits que representan el estado del número a medida que vamos leyendo. Para determinar, por ejemplo, el estado del número -12:

POKE2,66

El programa que determina el estado del número puede ser:

```
10 INPUT N:POKE2,0
20 IFN>100THENPOKE2,PEEK
   (2)OR128
30 IFN<0THENPOKE2,PEEK(2)
   OR64
40 IFN=0THENPOKE2,PEEK(2)
   OR32
50 IFN>1000THENPOKE2,PEEK
   (2)OR16
60 IFN=1THENPOKE2,PEEK(2)
   OR8
70IFN>10THENPOKE2,PEEK(2)
   OR4
80 IFN=INT(N)THENPOKE2,
   PEEK(2)OR2
90 IFN<>INT(N)THENPOKE2,
   PEEK(2)OR1
100 PRINTPEEK(2):GOTO10
```

Noten que las sentencias son todas iguales a diferencia del número con el cual se efectúa la OR. Es posible mejorar el programa a través de una subrutina formada por:

POKE2,PEEK(2)ORV

donde V representa el tipo de número que se está leyendo. (V toma los valores 128,64,32,....,1)

Antes de comenzar a explicar el programa, repasemos un poco las funciones lógicas AND y OR. Sus tablas de verdad son:

X	Y	X AND Y	X OR Y
0	0	0	0
0	1	0	1
1	0	0	1
1	1	1	1

Es decir que la función AND "será" 1 solamente cuando X e Y sean 1, mientras que en la OR esto ocurrirá cuando uno de los dos sea 1.

Empecemos a analizar el programa que determina el "estado" del número leído. En la línea 10 leemos éste y seteamos el contenido de la dirección $0002 a cero. A continuación comprobamos si se cumple la condición de que el número sea mayor que 100. Si ello ocurre ponemos el bit 7 (el cual representa ese hecho) a 1 efectuando una OR con el contenido actual de esa dirección (si no es mayor que 100 dejamos todo normal). Como estamos al principio, ésta será cero. Es decir efectuamos:

```
   00000000=($0002) (cuando
OR                 empezamos)
   10000000  (bit 7 en 1)
   --------
   10000000
```

Luego este resultado lo ponemos en esa dirección. Luego comprobamos si el número es menor que cero. Si ello ocurre, ponemos el bit 6 en 1 realizando una OR con el contenido actual de la dirección en cuestión que es, ahora, 10000000. De esta manera queda:

```
   10000000=($0002)
OR
   01000000
   --------
   11000000
```

el cual pasa a ser contenido temporal de la dirección $0002 (no olviden el POKE de cada una de las asignaciones PEEK(2)OR...). Habrán notado que el estado anterior sigue sin alterarse. Si hubiésemos utilizado una función AND en vez de la OR, se hubiese destruído todo estado actual del número. Sería un buen ejercicio comprobar esto último cambiando las OR por AND en nuestro programa.

Continuando con el análisis de nuestro programa, verificamos si el número es igual a cero. De ser así, ponemos el bit 5 a 1 haciendo:

```
   11000000=($0002)
OR
   00100000
   --------
   11100000
```

el cual pasa a ser el contenido de la dirección $0002. Seguimos con el mismo criterio hasta el final.

Ahora si deseamos detectar los diversos "estados" de un número, debemos utilizar la función AND ya que solamente nos importa si un bit determinado está en 1. La operación necesaria para llevar a cabo esto es:

PEEK(2)ANDX

donde X es el valor en decimal que deseamos detectar. Por ejemplo para chequear si el bit 7 está en 1, debemos efectuar:

```
    11100000=($0002)
AND
    10000000=128
    --------
    10000000
```

En este caso X debe ser 128. El resultado de esa operación nos dará el valor de X siempre y cuando el bit respectivo esté en 1. Qué pasaría si el bit a chequear fuese 0? El resultado sería:

```
    01100000=($0002)
AND
    10000000
    --------
    00000000
```

Comprueben ustedes que esto se verifica para cada uno de los bits y que, en caso de encontrarse un bit determinado en 1, se verificará:

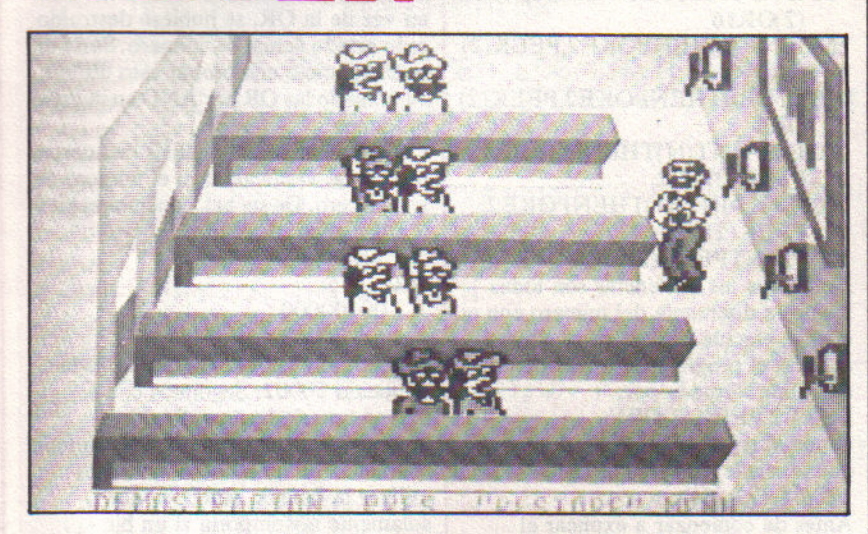
PEEK(2)ANDX=X

Prueben esto para cada uno de los bits de la dirección 2. Deben efectuar la operación AND entre ésta y:

10000000=128
01000000=64
00100000=32
00010000=16
00001000=8
00000100=4
00000010=2
00000001=1

Como corolario, observen que trasladar un bit hacia la derecha equivale a dividir por 2, mientras que trasladarlo hacia la izquierda equivale a multiplicarlo por dos.

TAPPER



Rating Total: A−
Creatividad: A−
Documentación: B+
Profundidad de juego: A−
Desafío: Difícil
Gráficos: B+
Valor en relación al costo: B
Mantiene el interés: A−
Computadora: Drean Commodore 64
Editor: Peek

En este juego el usuario es un atareado barman de un viejo salón del Oeste que debe servir bebida a sus clientes. En la primera imagen los clientes se alinean a lo largo de cuatro mostradores separados. El barman deberá llenar las jarras de las canillas que se encuentran a la izquierda y luego las deslizará a través del mostrador hacia los parroquianos.

El joystick controla los movimientos verticales entre los mostradores y los movimientos laterales desde el mostrador hasta las canillas. El usuario deberá ser muy rápido porque sino los clientes se impacientan y comienzan a aproximarse al barman. Si un cliente iracundo lo alcanza antes de que le haya servido su trago lo arrojará por el mostrador. Algunos clientes sólo se acercan para tomar sus bebidas y se retiran dejando lugar a nuevos clientes. Otros beben rápidamente sus tragos y deslizan las jarras vacías de vuelta hacia el pobre barman.

El hombrecito perderá la vida si una de las jarras cae al piso, o si el atareado empleado le arroja una bebida a un cliente que aún está bebiendo o que está mirando el show del lugar. Estas cosas no suceden hasta que el barman no haya recogido alguna propina dejada por algún cliente satisfecho.

Al comenzar el juego solamente se presentará un cliente por mostrador. Luego habrá dos y las cosas comenzarán a ponerse caóticas. Si el barman sobrevive avanzará hacia el triunfo. Luego, el infame "bandolero de los tragos" agitará cinco de las seis latas de Mountain Diew, después las mezclará y cuando él finalice, el barman deberá adivinar cuál es la que agitó. Si acierta, el usuario ganará 3000 puntos.

Tapper tiene otras variantes: el Jack Bar, el Punk Bar y el Space Bar. Mientras todas las canillas, en el Viejo Salón del Oeste y en el Jack Bar, están agrupadas en alguno de los costados; en el Punk y Space Bar están dispuestas entre los distintos mostradores. El bar Punk tiene dos canillas en la derecha en la base y dos más en la parte superior a la izquierda. El barman llega de una a otra a través de un corredor entre el segundo y tercer mostrador. La separación es aún peor en el Space Bar.

Desafortunadamente el usuario tiene que fijar las distintas instancias en orden sucesivo y difícilmente se alcancen las últimas dos.

Tapper es un juego lleno de emoción con gráficos espléndidos. Sigue un concepto verdaderamente original y mantiene el interés todo el tiempo. Un jugador puede servir los tragos sólo o dos jugadores pueden alternar turnos.

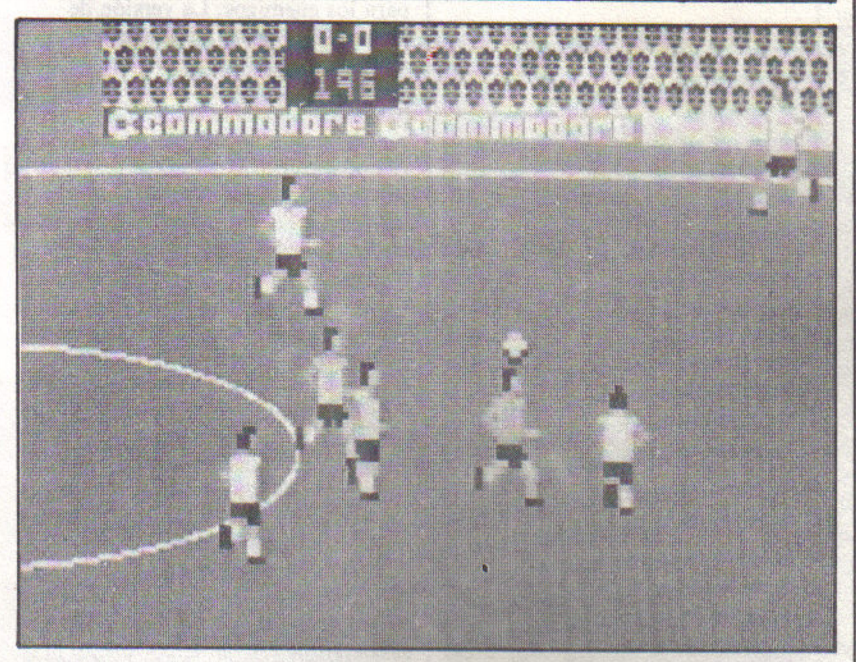
FUTBOL INTERNACIONAL

Rating Total: B
Creatividad: C
Documentación: C
Profundidad del juego: C
Desafío: Fácil
Gráficos: B−
Valor en relación al costo: B
Mantiene el interés: B
Computadora: Drean Commodore 64
Editor: Peek

Este juego es una de las simulaciones más realísticas de fútbol que existen para home computers. Commodore combina tres gráficos diferentes y una acción animada que asegura el entretenimiento de uno o dos jugadores.

El juego, que dura siete minutos divididos en dos tiempos, sigue todas las reglas del deporte. Cada equipo tiene seis jugadores y un arquero. El gráfico de los jugadores es desproporcionadamente grande en relación al campo de juego, pero su andar torpe y lento compensa su tamaño. El joystick controla un jugador por vez. El resto seguirá un patrón prefijado de acuerdo a la zona donde se hallen. Sus movimientos casuales raramente se relacionan con lo que acontece en el resto de la cancha y es muy poco lo que pueden hacer, salvo quedarse parados o frenar algún ataque del equipo contrario.

No se puede cambiar el control del jugador más cercano a la pelota pero si deliberadamente se saca al jugador bajo control fuera de la cancha el control cambiará. El jugador controlado estará indicado con una camiseta de un color más claro que las del resto del equipo. Dado que el jugador que lleva la pelota



siempre correrá más despacio que los defensores es preferible realizar pases abajo, es decir en la propia área. La pelota pica graciosamente proyectando una sombra tras de sí. Los jugadores pueden patear y también cabecear. El arquero puede controlarse una vez que el atacante se halla cerca de él. El usuario que en ese momento está defendiendo puede oprimir el botón del joystick para salvar el arco moviendo al arquero en dirección a la pelota.

Cuando la pelota se va fuera del área de juego se tira desde los laterales. El usuario dispone de unos pocos segundos para efectuar el tiro, pasado dicho lapso la computadora realiza el disparo automáticamente.

Fútbol internacional presenta muy buenos gráficos. El campo de juego está representado en tres imágenes diferentes que cambian lentamente proyectando la acción permanentemente. Los jugadores entran y salen de la pantalla constantemente. El usuario tiene una visión de la cancha desde el medio de una grada y ubicado a cierta altura, como si lo viera por televisión. Los jugadores ingresan y se retiran de la cancha al comenzar, terminar el partido, y en el entretiempo. El equipo ganador regresa una vez más al terminar el partido para recibir el trofeo. Estos simpáticos detalles hacen al juego aún más agradable.

Los nueve niveles de juego van desde un equipo deslucido a un equipo de campeones con una defensa y un ataque casi perfectos. Si se disfruta con los juegos que simulan deportes, con Fútbol Internacional se divertirá durante muchas horas.

ZAXXON

Rating Total: A
Creatividad: B
Documentación: C
Profundidad del juego: B
Desafío: Intermedio
Gráficos: A
Valor en relación al precio: A
Mantiene el interés: A
Computadora: Drean Commodore 64
Editor: Peek

Se trata de una versión para la Drean Commodore 64 de este juego que ya es un verdadero clásico. La gran calidad de sus gráficos lo distinguen de la mayoría de los trabajos de este tipo. Y a pesar de que los buenos gráficos actualmente no son tan inusuales, este juego sigue siendo realmente bueno.

El usuario vuela con su avión de guerra sobre una superficie plana de plataforma espacial. Sobre esa plataforma hay tanques de combustible, ametralladoras, misiles antiaéreos, fortalezas de piedra y otros armamentos futurísticos. El usuario deberá bombardear los tanques de combustible y las ametralladoras, mientras a vuelo rasante dispara sobre las fortalezas, que forman verdaderas vallas, y los misiles antiaéreos.

La habilidad y persistencia en el ataque posibilitarán que el usuario pase de la plataforma a una batalla espacial con otro avión y luego a una secuencia de plataforma de mayor dificultad y a una batalla robótica llamada Zaxxon. Una espléndida visión tridimensional distingue a Zaxxon de muchos juegos: se pueden controlar las posiciones verticales y horizontales sobre la plataforma espacial. Durante la batalla espacial, el usuario debe hacer coincidir ambas posiciones con los aviones que se aproximan para poder derribarlos con su artillería.

Los usuarios de Commodore pueden estar satisfechos ya que la versión de este juego es la mejor de todas las que hay en las home-computer o en las máquinas electrónicas. La enorme fortaleza es la más colorida y repleta de detalles gráficos que se haya concebido. Lo único criticable es el robot de Zaxxon, que es un tanto rudo e irreal.

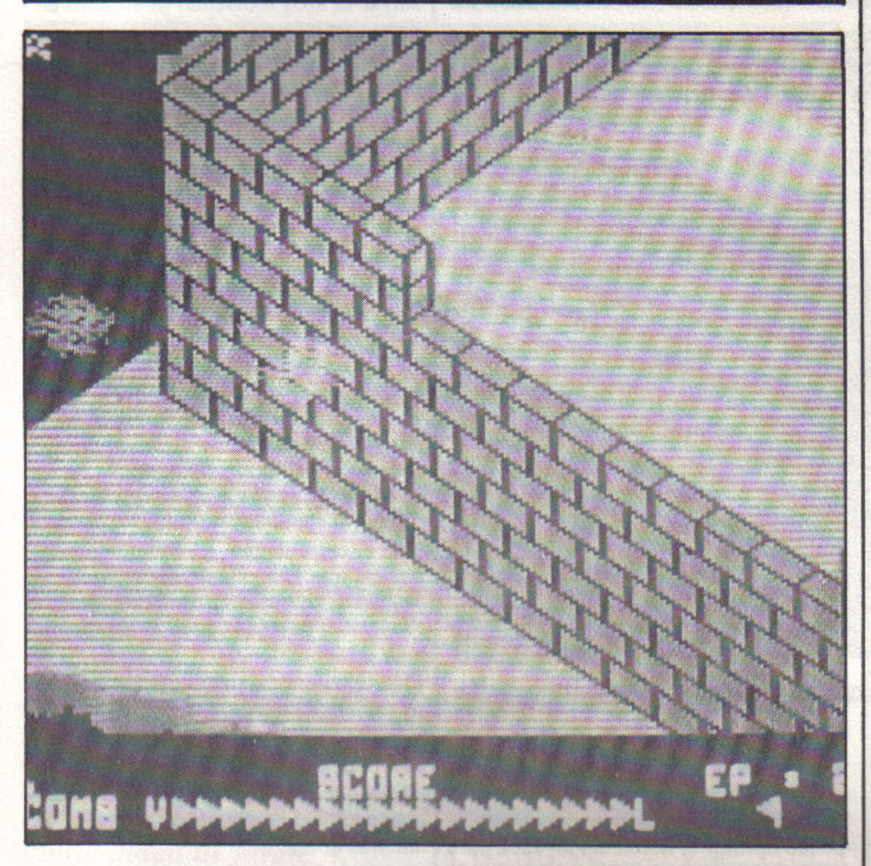
Los excelentes efectos de sonido imitan con bastante fidelidad a la versión Arcade (de las máquinas electrónicas) de este juego.

El trazado espacial está muy bien dibujado y el control de joystick está bien implementado. Las naves suben, descienden y se inclinan hacia derecha o izquierda exactamente como lo haría un aeroplano real. Los que recién se inician en el juego deberán observar el tablero de altitud a la izquierda de la pantalla o determinar la altura combinando diferentes datos, el lugar de proyección de las sombras de las naves y el golpe de los rayos laser. Una cosa que hace más simple a esta versión que la de Arcade es que el usuario no tendrá que preocuparse por los choques contra la cubierta de la fortaleza.

No se puede volar tan bajo.

Los aviones enemigos en la secuencia del espacio exterior son muy difíciles de alcanzar, ya que cambian de altura y se mueven de un lado al otro a medida que se acercan. Al igual que el avión identificado con el usuario, se ven más pequeños a medida que están a menor altura. Esta es una clave visual que ayudará a apuntar mejor hacia el objetivo.

Afortunadamente es tan difícil acertar



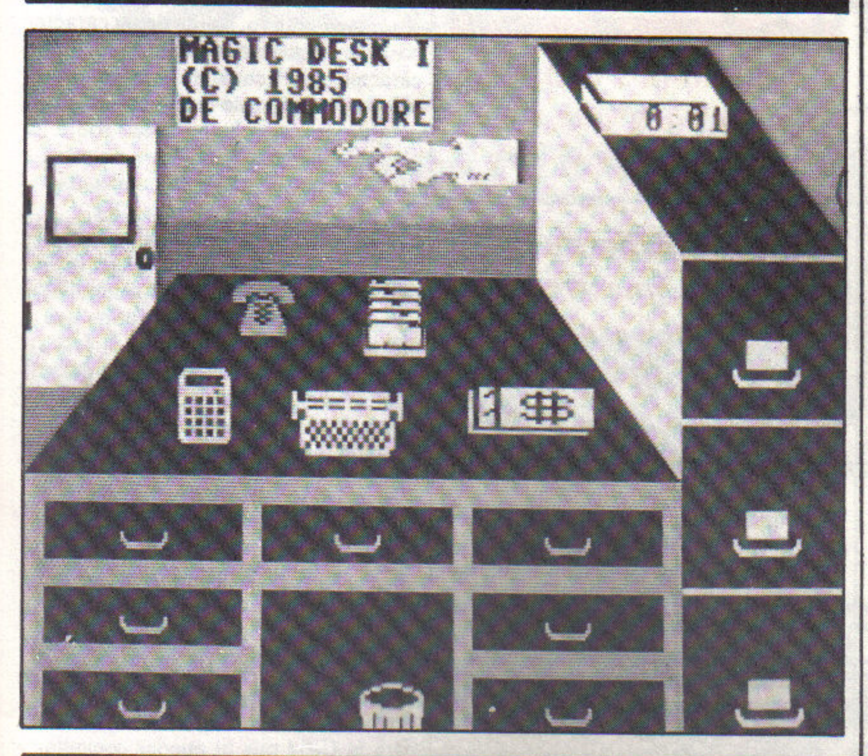
en el blanco tanto para el usuario como para los enemigos. La versión de Commodore posee un límite temporal en esta parte del juego, en vez de tener que esperar hasta destruir 20 naves para pasar a la siguiente secuencia, como sucede con la versión de Arcade.
El poderío del robot Zaxxon reside en la segunda fortaleza. Aberturas más angostas, más campos de fuerza, y misiles y ametralladoras, lo protegen de los ataques. Si el usuario consigue atravesar los seis o siete sectores de la fortaleza, la nave se detiene frente al robot y éste disparará lentos pero mortales misiles que siguen el calor hacia el avión del usuario. Para derrotarlo se deberán derribar esos misiles y acertar tres disparos en la plataforma de lanzamiento. Cuando Zaxxon es vencido comienza el juego en un nivel más difícil.
Esta versión de Zaxxon es muy buena en comparación con la de los juegos electrónicos porque es más fácil y alrededor de un veinte por ciento más lenta, y al mismo tiempo similar en cuanto a la calidad. Los excelentes gráficos son casi tan buenos como la primersa versión. Resumiendo, Zaxxon es uno de los mejores juegos del tipo de los de "disparos", que capturará la imaginación y mantendrá el interés.

MAGIC DESK

Rating Total: A—
Creatividad: B
Documentación: B
Gráficos: A
Valor en relación al precio: A
Computadora:
Drean Commodore 64
Editor: Peek

Una vez que el usuario carga el programa aparecerá en la pantalla un escritorio y una mano que le servirá para señalar el objeto deseado.
En el escritorio encontrará una máquina de escribir y también hallará un cesto de basura; un reloj y un armario de archivos, entre otras cosas.
A los otros objetos no se les da uso, para poder estandarizar el programa con cualquier periférico que Drean Commodore provee.
El reloj permite estar al tanto de la hora en cualquier momento que se desee siempre y cuando no se salga del programa. En el comienzo se sugiere como primera medida poner el reloj en hora. Luego hay que llevar la mano hasta el reloj y presionar el botón del joystick. Así aparecerá un recuadro alrededor del reloj que indicará que está habilitado para su uso. El usuario deberá tipear la hora deseada, colocando primero la hora y luego los minutos. Una vez terminada esta operación se deberá volver a presionar el botón que permitirá librarse de él.
Llevando la mano hacia la máquina de escribir y presionando el botón de disparo se habilita la máquina.
Accionando el botón se sale del modo de máquina de escribir y se debe guiar la mano hacia los marginadores.
Punteando con el botón dicha función se colocarán los márgenes donde el usuario lo desee. Para salir de ese modo, nuevamente se deberá presionar el botón.
El escritorio que allí aparece brinda la posibilidad de volver al comienzo para escoger otra función.
El cesto de basura permite deshacerse del texto que hasta ese momento se encuentra en la máquina de escribir y dejar un "papel" listo para volver a comenzar. Al presionar una vez el botón, aparecerá el papel encima del cesto. El usuario puede escoger dos caminos: 1) si presiona el botón, destruirá definitivamente el texto, 2) si mueve la palanca, rescata el texto; algo conveniente en caso de arrepentirse de destruirlo.
Una vez terminado el texto, se encontrará la posibilidad de imprimir el mismo. Si se lleva la mano a la impresora y se presiona el botón se pondrá directamente en funcionamiento la impresora, la cual le dará el texto. Si la impresora no está conectada o no tiene papel, aparecerá un símbolo en el dibujo del printer que indicará que el texto no sale por algún motivo.
Una vez concluído el texto se debe presionar el botón para parar el modo de impresión. De no hacerlo la impresora quedará inhabilitada para una próxima impresión.
Las funciones que brinda la máquina de escribir son tan variadas como las más modernas. Puede escribir tanto en mayúsculas como en minúsculas. Magic Desk ofrece tres cajones de

REVISION DE SOFTWARE



archivos y cada uno de ellos tiene una capacidad de diez carpetas. A su vez cada carpeta contiene una cantidad de diez hojas. Al llevar la mano hasta cualquiera de los tres archivos y presionar el botón de disparo se crea un nuevo archivo para la carta o documento que se tipeó o se quiere recuperar. Cuando se crea un nuevo archivo se debe poner en la carpeta deseada el nombre que se le asigna.

Al tipear el nombre y presionar el botón aparecerán luego en dicha carpeta las diez hojas correspondientes con el nombre. En el margen derecho de las opciones se encontrará una hoja en blanco. Esta da la posibilidad de pedir el contenido del texto de la hoja que se solicitó con anterioridad.

Puesto que se puede tener la necesidad de tener más archivos, se ha confeccionado este programa de manera tal que pueda iniciarse un disco virgen y llenarlo con más información.

Magic Desk es un verdadero procesador de textos. Ideal para oficinas, estudios y hogares.

CORREO - CONSULTAS

Ingresar datos

Jorge Lopez La Plata

Desearía saber si existe algún método para ingresar datos desde el teclado sin que aparezca el signo "?" de la sentencia INPUT

Una alternativa es utilizando la sentencia GET. Esta toma siempre, desde teclado o desde el diskette, un caracter a la vez. Pero vallamos al método en sí:

```
10 GETA$: IFA$= ""THEN10
20 IFA$=CHR$ (13) THEN 40
30 PRINTA$; :B$=B$+A$: GOTO
10
40 REM EN B$ ESTA EL STRING
INGRESADO
```

En la línea 10 esperamos hasta que se oprima alguna tecla.

Luego comprobamos si esta es la tecla de RETURN. Si no se orpimió esa tecla, entonces imprimimos el carácter seleccionado, formamos el string y volvemos a consultar el teclado. Si se oprimió la tecla de RETURN saltamos a la línea 40 y continuamos con el normal funcionamiento del programa (ya sea tomar alguna decisión acorde al dato ingresado, etc.)

Parámetro P

Edgardo Nuñez Santa Fé

Qué sucede si se omite el parámetro de posicionamiento (P) cuando se trabaja con archivos relativos?

Si se omite, el sistema operativo de la 1541 asume por default que se debe posicionar sobre el primer caracter del registro leído. De todas maneras te aconsejamos no omitir este parámetro ya que cuando se trabaja con esta clase de archivos se debe tener el máximo de cuidado posible.

Archivo

Susana Cuevas Rosario

Cuál es la forma correcta para abrir un archivo secuencial que será grabado varias veces sobre sí mismo?

La forma correcta de abrir un archivo secuencia es:

```
OPEN2,8,2," @ 0:NA,S,W"
```

donde NA es el nombre del archivo. De esta manera podrás grabar el mismo archivo varias veces sin que ocurra ningún error en la operación de la unidad 1541.

Números aleatorios

Graciela Miralles Capital

Cada vez que genero una serie de números aleatorios, éstos siempre se repiten. Cómo debo hacer para que ello no suceda?

Antes de comenzar a generar los números aleatorios inserta la siguiente línea:

```
X=RND(-TI)
```

luego utiliza tu RND. De todas maneras todas las funciones RND de las computadoras no son aleatorias cien por cien, sino que pseudoaleatorias es decir que, tarde o temprano, estas se repiten.

Reinicializar

Jorge Gonzalez Capital

Como puedo reinicializar la C-64 sin tener que utilizar el reset

El método de software para reinicializar la C-64 es ejecutnado una rutina que se encuentra en la dirección hexadecima E394 (58260 decimal). Si hacemos SYS 58260 se borrara la pantalla y aparecerá el mensaje de presentación de la C-64 cuando prendemos esta por primera vez. Todos los programas almacenados en la memoria RAM son borrados. En realidad lo que hace esta rutina es actualizar todos los punteros del Basic. Por ello el programa lo podemos recuperar tipeando:

```
POKE2050, 1: SYS42291
```

Sugerencias y pedidos

Les escribo con el fin de hacerles llegar algunas sugerencias y diversos pedidos. Mi nombre es Sergio F. Peña, tengo 18 años y soy técnico electrónico, y, como tal, me gustaría que publiquen una sección de hardware. Dentro de esta sección me sería de mucha utilidad la publicación del circuito Fast Load y la conexión que se debe realizar en el port para resetear la máquina.
Necesitaría, además, que me indiquen la función de cada pin del port del usuario.
Quisiera saber, en la salida AUDIO/VIDEO, cuáles pines corresponden a VIDEO y cuáles a AUDIO.
Para próximas ediciones sería interesante, por lo menos desde mi punto de vista, la publicación del esquima básico eléctrico del microprocesador 6510, con la correlatividad de los distintos pines, y el hardware que lo rodea.
Otra información que les agradecería muchísimo si la pudiesen publicar es sobre el moden telefónico (cómo está constituido y de qué forma trabaja).
Con respecto al software, me interesaría la publicación del set de instrucciones del 6510 para poder trabajar en lenguaje máquina.
Me disculpo por la gran cantidad de cosas que les pido, pero realmente todas son para mí de gran utilidad.
Les hago llegar mis saludos y felicitaciones ya que la publicación es muy buena especialmente para los que recién se inician en computación; eso sí: agréguenle algunas páginas más. SIGAN ASI!!

El material que tendrás para el número de abril será el set de instrucciones del microprocesador 6510, continuando con nuestra nota de assembler.
Respecto al circutio de Fast Load y del esquema electrónico del 6510 trataremos de hacértelo llegar a la brevedad (incluyendo, si es posible, algún circuito para moden telefónico compatible con la C-64).

Con respecto al reset de la máquina, las conexiones que se deben efectuar son muy fáciles. Se requiere cable de diámetro chico (1 mm aproximadamente) y un pulsador (a lo sumo se requieren de 2,5 australes para llevar a cabo el proyecto. Puedes utilizar el cable solo, aunque este método es muy peligroso).
Si observás la port del usuario verás que la plaqueta es doble faz. Sólo preocúpate por los pines 1 y 3 de la parte superior. Para localizarlos, debés contar los pines de derecha a izquierda estando de frente a la consola (como si fueses a escribir un programa). Paso seguido conectá el pulsador en esos pines. Cuando oprimas éste, verás como se resetea la máquina originando la impresión del mensaje que aparece cuando se prende la máquina por primera vez y, además, que todo programa en memoria se borrará.
Queremos decirte que no es nuestro objetivo que la publicación sea buena sólo para los que recién se inician sino que esperamos que sea buena para todos los que transitan por la informática, desde hace poco o mucho tiempo. De todas maneras agradecemos tus sugerencias y esperamos que sigas aportando más ideas.

Continuamos con esta sección para que los lectores planteen sus consultas y sugerencias. Para eso deberán escribir a nuestra redacción: Cerrito 1320, 1er. Piso, Buenos Aires (1010).

Subrutinas del area ROM.
Fabian Vali de Santa Fé

Me dirijo a ustedes con el propósito que me informen mejor sobre un tema al que se hace referencia en un artículo que habla del lenguaje assembler, en el ejemplar Nro. 1, acerca de las subrutinas incluidas en el área ROM. Me interesa saber cómo determinar dónde comienzan y de qué tipo son y si les es posible recomendar bibliografía sobre el tema.
Aprovecho la oportunidad para desearles suerte en la nueva empresa.

Esperamos, en próximos números, publicar en forma completa y detallada las subrutinas que utiliza el KERNAL de la C-64 (sistema operativo). De todas maneras podemos realizar un adelanto: La dirección de comienzo de dichas rutinas es la dirección hexadecimal FE2D. Aquí se hallan las más importantes como ser la rutina que inicializa el editor de pantalla, barrido del teclado, mensajes de control del sistema operativo, salida de un caracter por el canal actualmente abierto, ingreso de un caracter por el canal actualmente abierto, apertura de archivos, seteo del nombre de un archivo, ploteo sobre la pantalla, etcétera. Cada una de estas rutinas necesita, antes de ser ejecutada, sus parámetros respectivos. Por ejemplo, la rutina que saca un caracter hacia un periférico requiere que dicho caracter se encuentre en el acumulador y que se ejecute, antes, la rutina que efectúa la sentencia OPEN. Si ésta se omite, el KERNAL asumirá por default que el periférico es la pantalla. Por ejemplo, si deseamos escribir en la pantalla los caracteres "AB", debemos realizar:

```
LDA# $41      ;ASCII DE 'A'
JSR  $FFD2    ;LO SACAMOS
               POR PANTALLA
LDA# $42      ;ASCII DE 'B'
JSR  $FFD2    ;LO SACAMOS
               POR PANTALLA
RTS           ;FINALIZAMOS
```

De todas maneras, existe en la actualidad una bibliografía muy extensa para la C-64 que tú puedes consultar. Este tema en particular está explicado en el libro COMO PROGRAMAR SU C-64 (2do. Tomo, editorial PARANINFO). Aquí se describen cada una de las rutinas juntamente con sus respectivos parámetros para la correcta utilización de ellas.

consulta
Luis Echeverria Temperley

¿Cómo está formado un disco rígido?
El disco rígido es un medio preparado en superficies metálicas pulidas. Los discos se cubren con óxido magnético. Si es de tipo fijo, los discos están alojados permanentemente dentro de la unidad; no se puede sacar al medio ambiente sin deteriorar el dispositivo de almacenamiento masivo de información. Los que no son fijos se pueden intercambiar sin que se deterioren.

CP/M
Mabel Di Tomaso de Caseros

¿Pueden decirme qué es CP/M y para qué sirve?
La siglas CP/M significa Control Program/Monitor. Es un sistema operativo para computadoras cuyo microprocesador sea un Z80 o similares. Lo que hace CP/M es administrar los recursos de la computadora (memoria, periféricos, información y el procesador). La ventaja de disponer de CP/M es, que se pueden ejecutar cientos de programas escritos para este sistema operativo como ser DBaseII.CBASIC, etcétera.

√ Compútelo a su favor

En Argecint la Drean Commodore trae una tecla más: el respaldo.

DATASETTE

COMMODORE 16

JOYSTICK

PREMIUM

El respaldo, el service y el asesoramiento que puede ofrecer una empresa con 15 años, dedicados a la comercialización de todo lo que tenga que ver con el mundo de la computación. Argecint, El Súper Todo de Computación. Que pone a su alcance la Drean Commodore, con la línea más completa de accesorios, programas, periféricos... y con la financiación más cómoda de plaza y donde siempre encontrará un asesoramiento de expertos y una atención de amigos.

Drean Commodore y Argecint, la combinación perfecta.

PLANES UNICOS DE FINANCIACION - TARJETAS DE CREDITO

Casa Matriz: VENTURA BOSCH 7065 - Tel.: 641-0327/4892/3051 TELEX 17312 (ERSA) C.C. 8 Suc. 8 (1408) Cap. Fed.
Casa Central: AV. DE MAYO 1402 - Tel.: 37-4631 - Cap. Fed.
Agencia Trust: CARLOS PELLEGRINI Y CORRIENTES - Tel.: 35-5018/5029/0344 - Cap. Fed.
Agencia Belgrano: COMPUMARKET - AV. CABILDO 2869/71 - Tel.: 785-5241/4689 - Cap. Fed.
Agencia Flores: TRUST JOYERO - AV. RIVADAVIA 6687 - Tel.: 634-4639 - Cap. Fed.
Agencia Avellaneda: HIJOS DE G. ROSSI - AV. MITRE 660 - Tel. 201-5658 - Bs. As.
Sucursal Liniers: AV. RIVADAVIA 11332 (1408) - Tel.: 641-3088 - Cap. Fed.
Agencia Litoral: PEATONAL SAN MARTIN 2433 - Loc. 36 (3000) - Tel.: 25459 - STA. FE
Agencia Barrio Norte: AV. SANTA FE 2646 - Tel.: 84-8870 - Cap. Fed.
Agencia Computer Beach: AV. J.C. CHIOZZA 2872 (6561) - Tel.: 291 - SAN BERNARDO - BS. AS.

ARGECINT

el Super Todo de Computación

COMPUTADORES
PERIFERICOS
MAGNETICOS
MUEBLES
CINTAS
CASSETTES
ACCESORIOS
SUMINISTROS
FORMULARIOS
LAB. TECNICO
CURSOS DE COMPUTACION